



AFRL-RI-RS-TR-2022-065

DEEPWAVE: AUTOMATED RADIO SIGNAL AND PROTOCOL CLASSIFICATION THROUGH DEEP LEARNING FOR WAVEFORM VULNERABILITY DISCOVERY

UNIVERSITY AT BUFFALO

MAY 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2022-065 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

ELIZABETH SERENA BENTLEY
Work Unit Manager

/ S /

GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
MAY 2022		FINAL TECHNICAL REPORT		START DATE JANUARY 2020	END DATE DECEMBER 2021
4. TITLE AND SUBTITLE DEEPWAVE: AUTOMATED RADIO SIGNAL AND PROTOCOL CLASSIFICATION THROUGH DEEP LEARNING FOR WAVEFORM VULNERABILITY DISCOVERY					
5a. CONTRACT NUMBER N/A		5b. GRANT NUMBER FA8750-20-1-0501		5c. PROGRAM ELEMENT NUMBER 62788F	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R2XZ	
6. AUTHOR(S) Zhangyu Guan, Nicholas Mastronarde					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SUNY at Buffalo, Dept. of Electrical Engineering 230 Davis Hall Buffalo NY 14260				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITGB 525 Brooks Road Rome NY 13441-4505			10. SPONSOR/MONITOR'S ACRONYM(S) RI		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2022-065
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The research objective of this project is to enable a fundamental leap forward in the design, development, evaluation and experimentation of high-throughput wireless networking with guaranteed security in the presence of adversarial attacks. To this end, University at Buffalo (UB) and General Electric Aviation Systems (GEAS) propose to i) design new radio signal sensing and protocol classification techniques for automated discovery of the vulnerabilities of wireless systems; ii) simulate Unmanned Aerial System (UAS) networks in a contested, degraded, and operationally limited (CDO) environment in the UB's Airborne Networking and Communications (UB-ANC) Emulator using SwarmControl for dynamic network management and control of UAS swarms; and iii) integrate in-phase and quadrature (IQ) sample-level fidelity radio frequency (RF) simulation into the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) to provide a complete common operating picture for swarm operations.					
15. SUBJECT TERMS Swarm networking, autonomous swarms, artificial intelligence, spectrum sharing, packet boundary recognition					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		18. NUMBER OF PAGES 102
19a. NAME OF RESPONSIBLE PERSON ELIZABETH SERENA BENTLEY				19b. PHONE NUMBER (Include area code) N/A	

Table of Contents

1 Summary	1
2 Introduction	2
3 Methods, Assumptions & Procedures	6
3.1 Protocol-Agnostic Packet Boundary Recognition	6
3.1.1 Related Work	7
3.1.2 Packet Boundary Recognition Framework	8
3.1.3 Ground Truth Analysis	10
3.1.4 Data Collection	10
3.1.5 Performance Analysis	11
3.2 OSWireless Development	16
3.2.1 Motivation	16
3.2.2 Overall Objective and Design Approach	18
3.2.3 WiNAS Subplane Design	20
3.2.4 OaaS Subplane Design	25
3.2.5 Decomposition Engine Design	25
3.2.6 AlgoGen Engine Design	27
3.2.7 OSWireless Prototyping and Deployment	28
3.2.8 Experimental Evaluation	29
3.2.9 Discussions	35
3.2.10 Related Work	36
3.3 CloudRAFT: A Cloud-based Framework for Remote Experimentation for Mobile Networks	37
3.3.1 Motivation	37
3.3.2 CloudRAFT Framework	38
3.3.3 Enabling Techniques	40
3.3.4 Prototyping and Demonstrations	43
3.4 FlyBeam: Echo State Learning for Joint Flight and Beamforming Control in Wireless UAV Networks	45
3.4.1 Background	45
3.4.2 Related Work	46
3.4.3 System Model and Problem Formulation	47
3.4.4 FlyBeam Algorithm Design	49
3.4.5 Performance Evaluation	51
3.5 SwarmShare: Mobility Resilient Spectrum Sharing	55
3.5.1 Motivation	55
3.5.2 Related Work	56
3.5.3 System Model and Problem Formulation	57
3.5.4 Spectrum Coexistence Design	60
3.5.5 Performance Evaluation	64

3.6 RF-SITL Design.....	69
3.6.1 Background	69
3.6.2 Integration	70
3.6.3 Simulated COM Manager	72
3.6.4 Simulation Results	73
4 Results and Discussions.....	78
5 Conclusions	80
6 References	81
7 List of Acronyms	93

List of Figures

Figure 1: Software-defined packet recognition framework.....	7
Figure 2: A snapshot of the software-defined PBR testbed.....	11
Figure 3: Example PBR output using sensed signals (Welch's method). (a) Example of non-OTA data; (b) Example of sensed OTA data.	13
Figure 4: PBR rate for QPSK-modulated data streams with packet size of 1500, 2000, 4000 bytes and random size.	13
Figure 5: Analysis of modulation for KLD, 2000-byte packets.	14
Figure 6: Performance metrics of various test configurations. (a) PBR rate against transmission distance with bin size of 50 (left) and 100 (right) samples; (b) PBR rates with different packet sizes for entropy (left) and variance (right) approaches; and (c) PBR.....	14
Figure 7: Architecture of OSWireless SDN Controller.	18
Figure 8: Diagram of MathSpec Module.....	20
Figure 9: Diagram of NetTopo, QoSPara and ParaModel modules.	24
Figure 10: Overall architecture of OaaS Subplane.	24
Figure 11: Architecture of the software-defined testbed.....	28
Figure 12: Network topology for OSWireless testing over NeXT testbed.....	28
Figure 13: Intent specification for MSMH_Pow_ThrptRate_Max based on OSWireless APIs written in Python 3.0.....	30
Figure 14: Program files of the distributed cross-layer algorithmic specification automatically generated by OSWireless for network control problem MSMH_PowRate_ThrptMax.....	31
Figure 15: (a) Single-run running average and (b) multi-run average end-to-end throughput for MSMH_Pow_ThrptRate_Max.....	32
Figure 16: Instantaneous throughput of (a) Session 1 and (b) Session 2; (c), (d): Instantaneous transmission power of network control problem MSMH_PowRate_ThrptMax in Scenario 1.	33
Figure 17: Instantaneous (a) transmission rate and (b) end-to-end delay; (c) average end-to-end delay.	34
Figure 18: Optimized location for (a) Scenario 4, (b) Scenario 5 and (c) Scenario 6.	34
Figure 19: End-to-end throughput for (a) Scenario 4, (b) Scenario 5 and (c) Scenario 6.....	35
Figure 20: CloudRAFT: Cloud-based Remote Access for Wireless Testbeds.....	38
Figure 21: (a) Snapshot of USRP SDR deployment, (b) edge servers, and (c) ground robot vehicle.....	41
Figure 22: (a) Initializing VNC server and SSH tunnel; (b) initializing NoVNC server; and (c) NoVNC webpage.....	42
Figure 23: Demonstration of (a) point-to-point data transmission; (b) point-to-point data reception; and (c) online video streaming for remote experiment monitoring.	42

Figure 24: (a) Average sum capacity with different number of UAVs; (b) Capacity gain with different number of users.....	52
Figure 25: Average sum capacity with (a) varying flight altitude (b) varying number of blockages.....	53
Figure 26: (a) Computational complexity and (b) Communication overhead of FlyBeam. ..	54
Figure 27: Convergence of FlyBeam with different updating periods.	54
Figure 28: SwarmShare Spectrum Sharing Framework.....	60
Figure 29: (a) Snapshot of the testbed setup for threshold angle measurement; (b) Examples of the measurement results.....	64
Figure 30: Case study of power control based on SwarmShare. D#1[#2]: #1 is the UAV index, and #2 denotes the transmission power of the UAV in mW.....	66
Figure 31: Aggregate interference pdf with (a) 10 and (b) 20 UAVs; (c) Validation of data-driven predication of the SINR outage probability for the incumbent system.	66
Figure 32: Instantaneous capacity of the incumbent system and the UAV network with hovering UAVs. The violation probabilities are (a) 0.032, (b) 0.029 and (c) 0.021, respectively.....	67
Figure 33: Instantaneous capacity of the incumbent system and the UAV network with moving UAVs. The violation probabilities are (a) 0.023, (b) 0.019 and (c) 0.018, respectively.	68
Figure 34: Average capacity of the incumbent system and UAV networks with moving UAVs.	68
Figure 35: SINR outage probability vs UAV location reporting period.....	69
Figure 36: Example RF-SITL flowgraph with 3 transceivers and 3x3 channel blocks.....	70
Figure 37: Example Single Radio Flowgraph in RF-SITL.....	72
Figure 38: Scenario 1: Snapshot in AFSIM.....	74
Figure 39: Scenario 1: Packet transmissions between nodes.....	74
Figure 40: Scenario 2: Snapshot in AFSIM.....	75
Figure 41: Scenario 2: Packet transmissions between nodes.....	75
Figure 42: Scenario 3: Snapshot in AFSIM.....	76
Figure 43: Scenario 3: Packet transmissions between nodes.....	76
Figure 44: Scenario 4: Snapshot in AFSIM.....	77
Figure 45: Scenario 4: Packet transmissions between nodes.....	77

List of Tables

Table 1: Summary of Generated Datasets.....	12
---	----

1 Summary

The research objective of this project is to enable a fundamental leap forward in the design, development, evaluation and experimentation of high-throughput wireless networking with guaranteed security in the presence of adversarial attacks. To this end, University at Buffalo (UB) and General Electric Aviation Systems (GEAS) propose to i) design new radio signal sensing and protocol classification techniques for automated discovery of the vulnerabilities of wireless systems; ii) simulate Unmanned Aerial System (UAS) networks in a contested, degraded, and operationally limited (CDO) environment in the UB's Airborne Networking and Communications (UB-ANC) Emulator using SwarmControl for dynamic network management and control of UAS swarms; and iii) integrate in-phase and quadrature (IQ) sample-level fidelity radio frequency (RF) simulation into the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) to provide a complete common operating picture for swarm operations. Six major accomplishments have been achieved through this project: i) we designed a new framework for protocol-agnostic sensing for over-the air (OTA) RF signals, by taking packet boundary recognition (PBR) as an example; ii) we designed OSWireless, a new control plane for optimizing software-defined wireless networks; iii) we propose a cloud-based framework called CloudRAFT, based on which experimenters are allowed to remotely access and control experimental resources via public cloud Amazon Web Service (AWS) and share the resulting data and code via the cloud; iv) we proposed FlyBeam, a learning-based framework for joint flight and beamforming control in swarm UAV networks; v) we proposed *SwarmShare*, a mobility-resilient spectrum sharing framework for swarm UAV networking in the 6 GHz band; and vi) we collaborated with GE Aviation to integrate RF-SITL with their hardware-in-the-loop simulation environment comprising three M-100 flight controllers for simulated Unmanned Aerial Vehicles (UAVs) that act as independent agents within an AFSIM simulation framework. The research results have led to eleven (11) publications.

2 Introduction

This project focuses on wireless network security in the presence of advanced protocol-based denial-of-service (DoS) attacks and developing new radio signal sensing and protocol classification techniques based on deep learning theories to enable automated discovery of the vulnerabilities of wireless systems. Considering network scenarios where the attackers do not have access to either the transmitting or receiving nodes, the objective of this project is two-fold: (i) design and test experimentally radio signal sensing and protocol classification techniques for automated discovery of the vulnerabilities of wireless systems; (ii) understand to what extent existing wireless systems are secure in the presence of advanced DoS attacks. Toward this goal, this project proposes to develop DeepWave, a new, deep-learning-based radio signal sensing and protocol classification system, which can be used to recognize automatically the types of radio signals, unveil the hidden pattern of the signals, and further discover the vulnerabilities of the protocols adopted by the legitimate wireless systems, while without requiring the cooperation of the legitimate systems.

In the meantime, the Information Directorate of the Air Force Research Laboratory (AFRL/RI) has invested significantly in the development of a mission responsive cross-layer network protocol stack to support autonomous UAS swarm operation in contested, denied, and operationally limited (CDO) environments, with the recent AFRL program Cross-layer Heterogeneous Autonomous Resilient On-Demand Networks (CHARON) focusing on dynamic network management across the application, transport, and network layers, and previous efforts emphasizing cross-layer cognitive waveform technologies at the physical and medium access control layers. To fully exploit the mission-responsive cross-layer capabilities of CHARON, a key step is to address the requirements of the modeling, simulation, and emulation (MS&E) infrastructure that plays a critical role in transitioning novel technologies and results from the research community to demonstrations on hardware in mission relevant settings. Towards this goal, General Electric Aviation Systems (GEAS) has invested \$400k in 2019 for both CHARON and UB-ANC Emulator development. In this project, GEAS and UB propose to fully integrate these efforts by executing the following two critical elements: i) Simulate UAS networks in a CDO environment in the UB-ANC Emulator using Swarm Control for dynamic network management and control of UAS swarms; and ii) Integrate IQ-fidelity RF simulation into AFSIM to provide a complete common operating picture for swarm operations.

Through this project, the following six accomplishments have been achieved:

- **Protocol-Agnostic Sensing.** Radio-frequency (RF) sensing is a key technology for designing intelligent and secure wireless networks with high spectral efficiency and environment-aware adaptation capabilities. However, existing sensing techniques can extract only limited information from RF signals or assume that the RF signals are generated by certain known protocols. As a result, their applications are limited if proprietary protocols or encryption methods are adopted, or in environments subject to errors such as unintended interference. To address this challenge, in this project we study *protocol-agnostic cross-layer sensing*, where the objective is to extract high-layer protocol information from raw RF samples *without* any a priori knowledge of the protocols. First, we present a framework for protocol-agnostic sensing for over-the air (OTA) RF signals, by taking packet boundary recognition (PBR) as an example. The framework consists of three major components: *OTA Signal Generator*, *Agnostic RF Sink*, and *Ground Truth Generator*. Then, we develop a software-defined testbed using Universal Software Radio Peripheral (USRP) Software Defined Radios (SDRs), with eleven statistical algorithms implemented in the *Agnostic RF Sink*. These include distance correlation, Kullback-Leibler divergence, and cross-power spectral density, among others. Finally, we test the effectiveness of these statistical algorithms in PBR by generating a dataset of OTA RF samples, considering a wide variety of transmission parameters, including modulation type, transmission distance, and packet length. It is found that there is *no* single existing statistical algorithm that can achieve consistently high PBR rate. The parameters that affect most the PBR rate include modulation type and packet size, while, which is somewhat surprising, the PBR rate is barely affected by the transmission distance hence the received signal strength for most modulation schemes.

• **OSWireless.** In the current practice of wireless engineering, to optimize wireless networks engineers usually need to grapple simultaneously with network modeling, algorithm and protocol design, as well as their implementation on distributed edge nodes. This process is tedious and error prone. In this report, we attempt to address this challenge by designing OSWireless, a new control plane for optimizing software-defined wireless networks. At the core of OSWireless is the virtualization of four control plane functionalities, including intent, mathematical, algorithmic and forwarding specifications, and then provide them as a service to network engineers. To this end, we design two new sub-planes for the control plane: Wireless Network Abstraction Specification (WiNAS) Sub-plane and Optimization-as-a-Service (OaaS) Sub-plane. The former converts intent specifications defined using high-level application programming interface (APIs) to the corresponding mathematical specifications, and the latter generates automatically operational (possibly distributed) algorithmic specifications. We prototype OSWireless and deploy it over NeXT, a newly developed software-defined experimentation testbed, and showcase the automated control program generation capability of OSWireless and the optimality of the resulting programs considering a variety of network control problems. Through developing OSWireless and making it open source, we hope to accelerate research towards future zero-touch software-defined wireless networks with reduced management complexity.

- **CloudRAFT.** In this project we explore new techniques that can enable open remote experimentation for mobile networks. We first propose a cloud-based framework called CloudRAFT, based on which experimenters are allowed to remotely access and control experimental resources via public cloud Amazon Web Services (AWS) and share the resulting data and code via the cloud. Then, we discuss the enabling techniques for CloudRAFT, including Amazon serverless service, Virtual Network Computing (VNC)-based remote command line, and Websocket-based real time communications, among others. Finally, we showcase the application of these techniques in enabling remote access to UB NeXT, a software-defined testbed that has been developed at University at Buffalo for wireless mobile network modeling, optimization and deployment. This work verifies the feasibility of accessing, controlling and sharing wireless testbeds through a remote public cloud.
- **FlyBeam.** This work aims at designing high-data-rate swarm unmanned aerial vehicle (UAV) networks with distributed beamforming capabilities. The primary challenge is that the beamforming gain in swarm UAV networks is highly affected by the UAVs' flight altitude, their movements and the resulting intermittent link blockages, as well as the availability of channel state information (CSI) at individual UAVs. To address this challenge, we propose FlyBeam, a learning-based framework for joint flight and beamforming control in swarm UAV networks. We first present a mathematical formulation of the control problem with the objective of maximizing the throughput of swarm UAV networks by jointly controlling the flight and distributed beamforming of UAVs. Then, a distributed solution algorithm is designed based on a combination of Echo State Network learning and online reinforcement learning. The former is adopted to approximate the utility function for individual UAVs based on online measurements, by jointly considering the unknown blockage dynamics and other factors that affect the beamforming gain. The latter is used to guide the exploitation and exploration in FlyBeam. The effectiveness of FlyBeam is evaluated through an extensive simulation campaign. Results indicate that significant (up to 450%) beamforming gain can be achieved by FlyBeam. We also investigate the effects of blockages and UAV flight altitude on the beamforming gain. It is found that, which is somewhat surprising, higher (rather than lower) beamforming gain can be achieved by FlyBeam with denser blockages in swarm UAV networks.

- **SwarmShare.** In this project, we propose *SwarmShare*, a mobility-resilient spectrum sharing framework for swarm UAV networking in the 6 GHz band. We first present a mathematical formulation of the *SwarmShare* problem, where the objective is to maximize the spectral efficiency of the UAV network by jointly controlling the flight and transmission power of the UAVs and their association with the ground users, under the interference constraints of the incumbent system. We find that there are no closed-form mathematical models that can be used to characterize the statistical behaviors of the aggregate interference from the UAVs to the incumbent system. Then we propose a data-driven three-phase spectrum sharing approach, including *Initial Power Enforcement*, *Offline-dataset Guided Online Power Adaptation*, and *Reinforcement Learning-based UAV Optimization*. We validate the effectiveness of *SwarmShare* through an extensive simulation campaign. Results indicate that, based on *SwarmShare*, the aggregate interference from the UAVs to the incumbent system can be effectively controlled below the target level *without* requiring the real-time cross-system channel state information. The mobility resilience of *SwarmShare* is also validated in coexisting networks with *no* precise UAV location information.
- **RF-SITL.** RF-SITL is a software-in-the-loop (SITL) IQ-fidelity software-defined channel emulator implemented in GNU Radio. In this project, we collaborated with GE Aviation to integrate RF-SITL with their hardware-in-the-loop simulation environment comprising three M-100 flight controllers for simulated UAVs that act as independent agents within an AFSIM simulation framework. The integration was validated in four progressively more sophisticated AFSIM scenarios demonstrating: 1) successful communication between two nearby UAVs; 2) failed communication between two distant UAVs; 3) multi-hop communication through three relay UAVs; and 4) multi-hop communication with an interferer that disrupts communication among relays.

3 Methods, Assumptions & Procedures

3.1 Protocol-Agnostic Packet Boundary Recognition

Radio-frequency (RF) sensing is a key technology to design next-generation intelligent and secure wireless networks with higher spectral efficiency and better resilience against adversarial attacks. In the past decades, RF sensing has attracted significant research interest, focusing on cognitive radio [1], spectrum coexistence [2], localization and tracking [3], and detection of adversarial interference [4], among others. This article studies a new application of RF sensing, which we call *protocol-agnostic cross-layer sensing*, in non-cooperative environments with *limited or no* a priori knowledge of the protocols adopted by the wireless networks.

Why Protocol-Agnostic Cross-Layer Sensing? Different from the RF sensing at the physical layer, where the primary objective is to detect the presence and strength of radio signals [5,6], cross-layer RF sensing aims to discover a richer set of signal features at higher protocol layers, and hence to unveil the hidden traffic pattern and enable more sophisticated adaptation to the dynamic spectrum environments or launch more effective denial of service (DoS) efforts. For example, it is discussed in [4] that, compared to traditional continuous noise interference, significantly more effective interference attempts can be launched against a network if adequate knowledge of the network’s transmission protocols can be obtained. Towards this end, the low-level time-domain information of the control packets of a known protocol such as timing and length can be used to implement protocol-aware interference to target those control packets [4]. While existing research has shown great potential of cross-layer sensing, there are still several challenges to address. First, the quality of the received RF signals, in terms of signal strength and signal-to-noise ratio (SNR), is highly affected by path loss, fading, shadowing, noise, and other factors. As a result, it is difficult to extract any useful high-layer protocol information without successful decoding of the packets because of the poor-quality RF signals. Moreover, it is also challenging to perform cross-layer sensing in those low-power IoT systems [7,8] such as Bluetooth, LoRaWAN and SigFox, where spread-spectrum and frequency hopping techniques have been utilized for interference avoidance in high-density RF environments [7].

Second, as more spectrum resources are made open for unlicensed use, different wireless systems may need to co-locate on the same frequency band while operating different, possibly proprietary protocols. For example, the ISM bands (USA: 915 MHz, Europe: 868 MHz) are home to many low-power, wide area network (LPWAN) technologies such as SigFox and LoRaWAN [9]. SigFox is an Ultra-Narrowband (UNB) communication scheme based on proprietary MAC-layer protocol, which ensures a high degree of network security by limiting the over-the-air (OTA) activity of devices within a network in addition to non-OTA activation of end devices [10]. In such scenarios, higher-layer protocol information cannot be sensed by decoding the packet header information because of the proprietary protocols.

To address the above challenges, a key step is to understand through rigorous experiments the limitations of existing approaches for protocol-agnostic RF sensing. In this work we make an initial step towards this direction, taking packet boundary recognition (PBR) at the link layer based on raw RF samples at the physical layer as an example. We consider PBR because it is an

enabling technique for collecting various upper-layer protocol characteristics of wireless networks, e.g., packet duration, timing, periodicity as well as type (e.g., control vs data). A wide set of new sensing-based applications can be enabled by our work, including intelligent spectrum access, wireless network traffic and non-cooperative interference detection, among others, through enhanced environmental awareness and protocol identification.

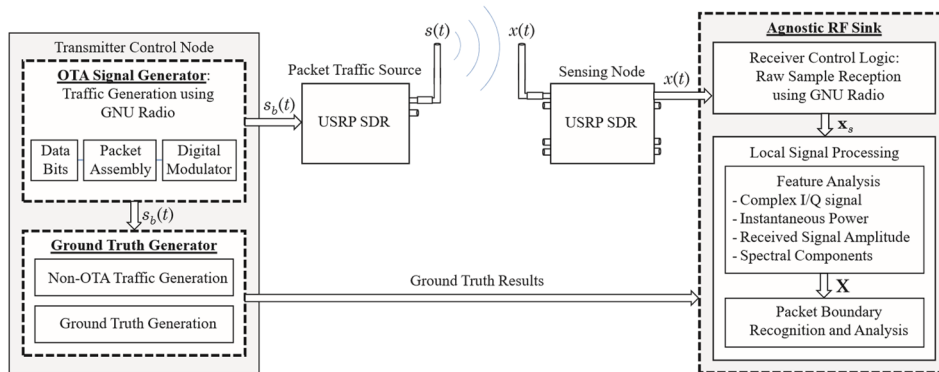


Figure 1: Software-defined packet recognition framework.

3.1.1 Related Work

RF sensing has attracted significant research attention with a sizable and growing body of literature [1–3,6,11–14]. For example, the authors of [11] prove the ability to classify traffic by protocol according to only a fraction of packets within a stream, assuming packet size, timing, and received power levels to be known. Similarly, in [12] Dempsey et al. sense and classify different types of packets generated by a known set of network and transport layer protocols, assuming packet size, timing and other lower-layer packet information to be known. A wideband compressive sampling method is introduced in [15] to detect wireless signals in frequency hopping networks. The authors of [16] propose a simplified iterative compressive sensing approach to detect spectrum holes in ultra-wideband wireless UAV networks. Datta et al. compare in [17] the efficiency of cooperative sensing methods for spectrum detection by considering different fading models. In [18], the authors use a spectral correlation function to extract second-order statistic information of the RF samples to detect spectrum occupants. Please refer to [6,19] and references therein for a good survey of the main results in this field. Different from these works, where the physical layer information is usually assumed to be known, in this project we focus on protocol-agnostic RF sensing with *limited or no* a priori knowledge of the signals.

The use of physical layer characteristics in protocol-agnostic applications has been most explored in the context of modulation recognition (MR) [20–24]. For example, a discrete wavelet two-stage neural network group is introduced in [20] to use time-frequency features for automatic MR. In [21,22], O’Shea et al. show that convolutional neural networks (CNNs) trained on OTA

radio signals can be an effective alternative to traditional feature-based MR. Zhang et al. propose in [23] to utilize frequency-domain information of the radio signals as input to a CNN for MR. Readers are referred to [24] and references therein for a good survey in this area. Different from MR, in this work we focus on a new application of protocol agnostic sensing for packet boundary recognition.

3.1.2 Packet Boundary Recognition Framework

The framework of the packet recognition system is illustrated in Fig. 1, where there are three major components: OTA Signal Generator, Agnostic RF Sink, and Ground Truth Generator. In the OTA Signal Generator, the passband source signal $s(t)$ is first generated by modulating baseband signal $sb(t)$ to a carrier frequency f_c , i.e., $s(t) = \Re\{sb(t)e^{j2\pi f_c t}\}$ and then sent to the SDR front-end of the OTA Signal Generator for over-the-air transmissions. At the same time, the baseband signal $sb(t)$ is also recorded as a non-OTA traffic sequence in the Ground Truth Generator to extract ground truth packet boundary information.

The signals received by the Agnostic RF Sink, denoted as $x(t)$, can be given as $x(t) = s(t) * h(t) + \eta(t)$, where $*$ is the convolution operation, $h(t)$ is the channel gain, and $\eta(t)$ represents noise introduced by the channel. The continuous signal $x(t)$ is first quantized by the Receiver Control Logic in the RF Agnostic Sink into a stream of discrete samples represented as a vector $x = [x_1, x_2, \dots, x_S]$, where S is the length of the quantized stream in samples. The resulting sample stream is then further divided into a set of consecutive bins each of size B in the Local Signal Processing block.

Denote the resulting set of bins as $\mathbf{x} = \{x_1, x_2, \dots, x_\beta\}$, in which $\beta = \lfloor \frac{S}{B} \rfloor$ and $\lfloor \cdot \rfloor$ is the integer floor operation. The obtained bins $\mathbf{x}$ are finally used by the Agnostic RF Sink for feature extraction and PBR analysis. The analysis results are then evaluated by comparison with the ground truths captured by the Ground Truth Generator. This will be further discussed later in this section. Our objective is to understand the limitations of existing statistical methods for protocol-agnostic sensing. To this end, eleven statistical PBR approaches have been implemented in the Local Signal Processing block of the Agnostic RF Sink. These approaches include cross-correlation, variance, covariance, entropy, cross-entropy, Pearson's correlation, Fisher's method, Kullback-Leibler Divergence (KLD), distance correlation, Welch's method, and cross-spectral density (CSD). These approaches have been selected to test, in a comparative manner, their potential for discovering the target patterns of the test data by considering a wide range of experimental configurations. Next, we discuss two representative categories of PBR analysis approaches to illustrate the experimental process.

Entropy, Cross-Entropy, and Relative Entropy Approaches. Entropy is a notion from information theory, taken as a measure of information, and has been applied to pattern recognition problems [25]. For PBR, sample bins containing inter-packet gaps are expected to exhibit different levels of entropy than bins without any gaps. In this work, Shannon entropy is considered. Given a single sample bin $X_m \in \mathbf{X}$, the corresponding entropy, denoted $H(X_m)$, can be calculated as

$$H(X_m) = - \sum_{i=1}^B P(x_{m,i}) \log_2 P(x_{m,i}) \quad (1)$$

where $P(x_{m,i})$ is the probability of sample $x_{m,i}$ in bin X_m .

Cross-entropy can be calculated similarly. For adjacent sequential bins $X_m, X_n \in \mathbf{X}$, their cross-entropy, denoted $H(X_m, X_n)$, can be calculated as_B

$$H(X_m, X_n) = - \sum_{i=1}^B P(x_{m,i}) \log_2 P(x_{n,i}) \quad (2)$$

Kullback-Leibler Divergence (KLD), or relative entropy, can be used to find the change in entropy based on statistical similarity of the two bins [26]. For sample bins $X_m, X_n \in \mathbf{X}$, the KLD, denoted $KL_{m,n}$ can be computed as

$$KL_{m,n} = H(X_m, X_n) - H(X_m), \quad (3)$$

where $H(X_m)$ and $H(X_m, X_n)$ represent the entropy and cross-entropy defined in (1) and (2), respectively. In this work, entropy and cross-entropy are used in feature extraction from RF data to exploit the difference in the distribution of in-packet samples from samples containing the inter-packet gap. For two identical sample bins, the computed KLD will be zero as there is no relative change in entropy.

Welch's (Periodogram) Method. This method is based on short-time spectral density estimation to show frequency domain information for each sample bin. These periodograms generated from overlapping windows within each sample bin using the discrete Fourier transform (DFT). The n -th sample in the m -th window of a sample bin, denoted as $x_m(n)$, can be expressed as:

$$x_m(n) \triangleq w(n)x(n + mR), \quad (4)$$

where $m = 0, 1, \dots, M-1, n = 0, 1, \dots, N-1$ with M being the number of windows per bin and N the number of samples per window; $x(n+mR)$ is the $(n+mR)$ -th sample of the bin, with R representing the window hop size (e.g., $R = 0.5$ allows 50% window overlap) and w is the windowing function. In this work, a Hann (raised cosine) window function has been used to prevent window edge discontinuities.

Then, the periodogram, denoted as $P_m(\omega)$ for the m -th window of a sample bin at frequency ω , can be written as

$$P_m(\omega) = \frac{1}{N} |FFT_N(\mathbf{x}_m)|^2 \triangleq \frac{1}{N} \left| \sum_{n=0}^{N-1} x_m(n) e^{-\frac{j2\pi n}{N}} \right|^2, \quad (5)$$

where $\mathbf{x}$ -th window with $x_m(n)$ defined in (4). For each sample bin, let $S(\omega)$ represent the average of the resulting periodograms, then $S(\omega) \triangleq \frac{1}{M} \sum_{m=0}^{M-1} P_m(\omega)$, where $P_m(\omega)$ is the periodogram calculated in (5). Since this method takes a single sample bin as input, the target feature is defined as the magnitude of the peak frequency component of each sample bin.

3.1.3 Ground Truth Analysis

The PBR results obtained by the statistical approaches will be compared against two types of ground truth information: the number of packets in each transmission denoted as G and the sample index of g -th packet boundary denoted as g_{smp} . The number of packets in each transmission, i.e., G , is known from the time of transmission and will be used to determine the overall PBR rate for each of methods discussed above. Denote the PBR rate as $\Gamma = \frac{G_{prd}}{G}$, where G_{prd} is the predicted number of packets in the stream.

To verify individual packet boundary locations, we first determine the sample index g_{smp} and bin index g_{bin} of the g -th calculated boundary and then compare them to the ground truth boundary locations. With a fixed transmission bit rate R_{bit}^{TX} and receive sample rate R_{smp}^{RX} the expected location of the g -th packet boundary can be predicted as, in the case of uniform packet length L_{byte} , $g_{smp} = \hat{g}_{smp} + 8gL_{byte} * \frac{R_{smp}^{RX}}{R_{smp}^{TX}}$, where $\hat{g}_{smp}$ represents the index of the starting sample of the first packet within the collected RF data.

3.1.4 Data Collection

Based on the PBR framework, a testbed has been developed using USRP software-defined radios. Figure 2 shows a snapshot of the testbed, where the transmitter and receiver are implemented using USRP N210 and B210, respectively. The control host is a Dell Latitude 5491 laptop, which implements the *OTA Signal Generator* and *Agnostic RF Sink* as customized implementations in GNU Radio, and *Ground Truth Generator* as functions in Python.

Two categories of data have been generated, based on the length of the modulated packets: fixed-length packet streams, to investigate how different packet sizes are detected, and random packet streams, to model more realistic network behaviors. The former consists of uniform packets of lengths of 1500, 2000, or 4000 bytes. The latter consists of packets with lengths determined by a known pseudorandom sequence, with a minimum length of 48 bytes and a maximum length of 4080 bytes. For each type of stream, five of the most widely-used digital modulations have been considered, including BPSK, QPSK, GMSK, 8-PSK, and 16QAM. The generated samples are first saved as arrays of complex *Non-OTA* data and sent to the *Ground Truth Generator* prior to OTA transmission.

On the receiver side, the OTA data is collected and preprocessed for PBR analysis. The data is first preprocessed, including quantization and binning of radio signals into vectors of sizes 50 and 100 samples. Each of the preprocessed datasets is then used to further generate three feature sets consisting of complex I/Q data, received signal amplitude, and short-time frequency spectrum. In total, 100 datasets of OTA radio signals have been collected, each consisting of around 3,000,000 samples, as well as 20 sets of non-OTA data each containing between 400,000 and 1,600,000 samples. The collected datasets are summarized in Table 1.

3.1.5 Performance Analysis

In this section, we evaluate the PBR approaches by comparing them against ground truths in terms of PBR rate and accuracy in a variety of conditions. The PBR rate refers to the percentage of packets detected successfully, while the PBR accuracy is defined as the distance between the predicted and the ground truth packet boundaries. Next, we study two PBR examples for non-OTA and OTA data in Figs. 3(a) and 3(b), respectively.

Figure 3(a) shows an example of the PBR results for Welch's method on non-OTA data with packet size of 2000 bytes and modulation types of BPSK (top), GMSK (middle) and 16-QAM (bottom). The output values of Welch's Method are proportional to the frequency content of the baseband signal. In Fig. 3(a), it can be found that the output values of Welch's Method range from 0 to 3. By comparing with the ground truth, we found that each of the major peaks in the three plots corresponds to a packet boundary, which verifies the effectiveness of Welch's Method in PBR.

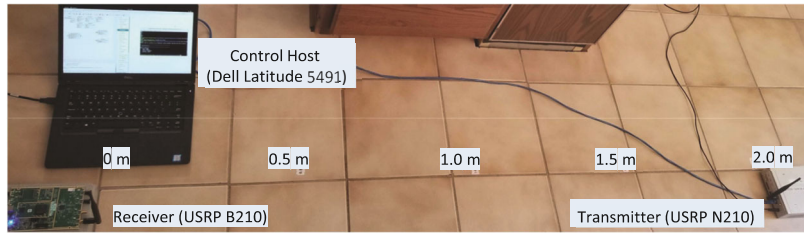


Figure 2: A snapshot of the software-defined PBR testbed.

Table 1: Summary of Generated Datasets.

Modulation	Stream Type	Packet Sizes (Bytes)	Distance (m)
BPSK	OTA (Streams 1-20)	1500 (Streams 1-5)	0.5 - 2.5
		2000 (Streams 6-10)	0.5 - 2.5
		4000 (Streams 11-15)	0.5 - 2.5
		Random (Streams 16-20)	0.5 - 2.5
	Non-OTA (Streams 101-104)	1500	—
		2000	—
		4000	—
		Random	—
GMSK	OTA (Streams 21-40)	1500 (Streams 21-25)	0.5 - 2.5
		2000 (Streams 26-30)	0.5 - 2.5
		4000 (Streams 31-35)	0.5 - 2.5
		Random (Streams 36-40)	0.5 - 2.5
	Non-OTA (Streams 105-108)	1500	—
		2000	—
		4000	—
		Random	—
8-PSK	OTA (Streams 41-60)	1500 (Streams 41-45)	0.5 - 2.5
		2000 (Streams 46-50)	0.5 - 2.5
		4000 (Streams 51-55)	0.5 - 2.5
		Random (Streams 56-60)	0.5 - 2.5
	Non-OTA (Streams 109-112)	1500	—
		2000	—
		4000	—
		Random	—
16-QAM	OTA (Streams 61-80)	1500 (Streams 61-65)	0.5 - 2.5
		2000 (Streams 66-70)	0.5 - 2.5
		4000 (Streams 71-75)	0.5 - 2.5
		Random (Streams 76-80)	0.5 - 2.5
	Non-OTA (Streams 113-116)	1500	—
		2000	—
		4000	—
		Random	—
QPSK	OTA (Streams 81-100)	1500 (Streams 81-85)	0.5 - 2.5
		2000 (Streams 86-90)	0.5 - 2.5
		4000 (Streams 91-95)	0.5 - 2.5
		Random (Streams 96-100)	0.5 - 2.5
	Non-OTA (Streams 117-120)	1500	—
		2000	—
		4000	—
		Random	—

The corresponding results for OTA data is reported in Fig. 3(b). In the test cases, the output values of Welch's method range from 0 to 16. Compared to non-OTA results in Fig. 3(a), the

clarity of the peak values has been somewhat degraded by channel effects inherent to OTA transmissions.

PBR Rate Analysis. In the following three experiments, we study the PBR rate of the 11 approaches. The results are reported in Figs. 4, 5, and 6. In Fig. 4, we plot the PBR rate for QPSK modulated packets of size 1500, 2000, 4000 bytes as well as random size. The bin size has been set to 100 samples. The results are obtained by averaging over 5 transmission ranges as shown in Fig. 2. It can be seen that the PBR rate ranges from 30% to 99% with different parameters, and that in general, a higher PBR rate can be achieved with longer packets. For example, with packet size of 1500 bytes, as shown in Fig. 4, the most accurate method is KLD, which can detect 76% of packets. The corresponding PBR rate is 99% for packet sizes of both 2000 and 4000 bytes. Similarly, a very good PBR performance can also be achieved by the distance correlation approach and variance approach, with a PBR rate of 97% for both approaches with packet size of 4000 bytes.

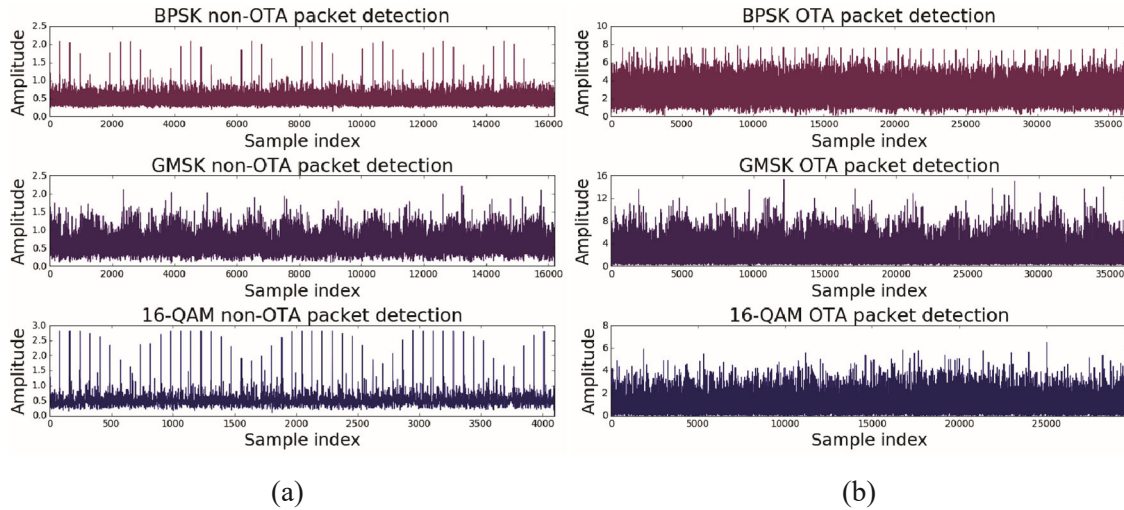


Figure 3: Example PBR output using sensed signals (Welch's method). (a) Example of non-OTA data; (b) Example of sensed OTA data.

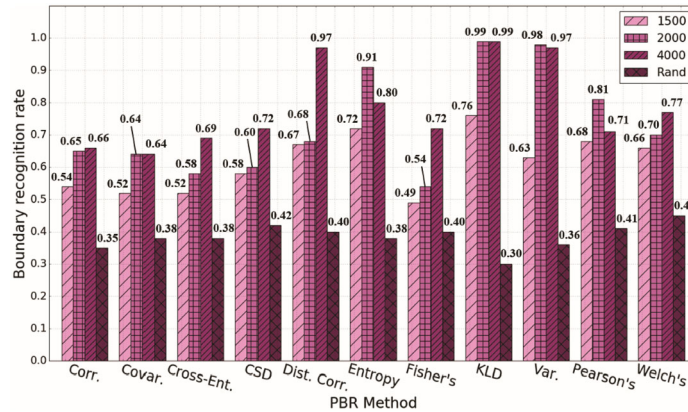


Figure 4: PBR rate for QPSK-modulated data streams with packet size of 1500, 2000, 4000 bytes and random size.

From Fig. 4, it can also be found that the PBR rate is significantly degraded for all 11 approaches when the packets are generated with random length. For example, the PBR rate is only 30%, 35%, 36%, 38% and 40% for the KLD, cross-correlation, variance, covariance, and distance correlation approaches, respectively. The most accurate method for random-sized packets is Welch's method, which can detect 45% of packet boundaries. This experiment implies that i) with the tested existing statistical approaches, we can expect PBR rates between 30% and 100% in real wireless networks, where packets are usually generated according to a fixed set of packet lengths; and ii) in the presence of unintended interference, it can be an effective security measure to randomize packet lengths in order to reduce the probability of detection.

In Fig. 5, we plot the PBR rate of the KLD approach considering 5 modulations and packets of sizes of 1500, 2000, and 4000 bytes as well as random size. The bin size has been set to 100 samples. We consider the KLD approach in this experiment because it achieves the highest PBR rate in most of the tested cases. It can be seen that it can achieve a PBR rate of 87%, 87%, 70%, and 99% with fixed packet lengths for BPSK, 8-PSK, 16-QAM, and QPSK modulations, respectively. The corresponding PBR rate with random packet lengths is 41%, 43%, 45%, and

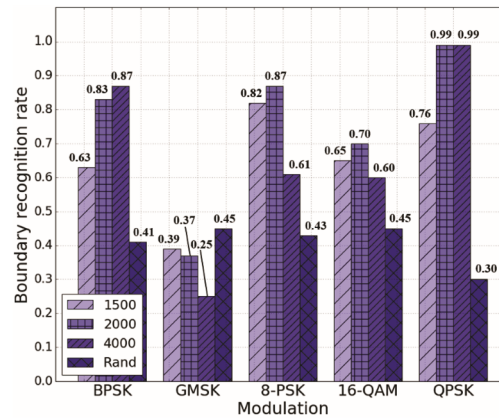


Figure 5: Analysis of modulation for KLD, 2000-byte packets.

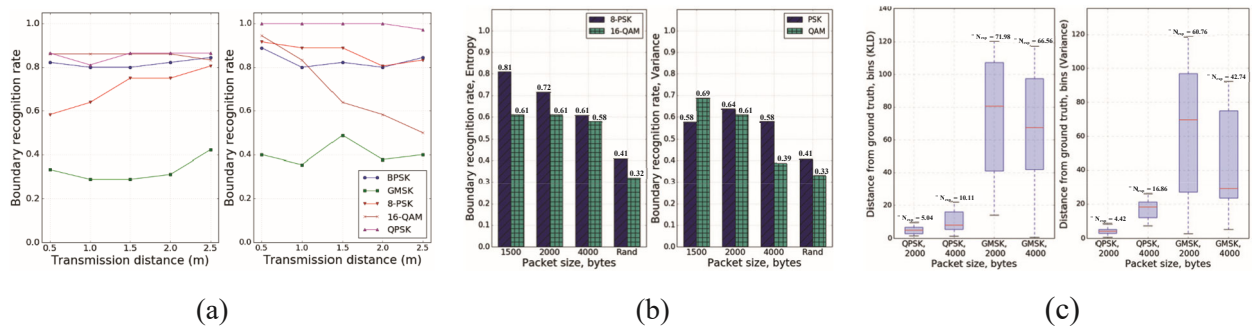


Figure 6: Performance metrics of various test configurations. (a) PBR rate against transmission distance with bin size of 50 (left) and 100 (right) samples; (b) PBR rates with different packet sizes for entropy (left) and variance (right) approaches; and (c) PBR

30%. The exception is GMSK, for which a PBR rate of less than 50% can be achieved even with fixed packet lengths; for this modulation, according to our experiments the most effective PBR approaches are Welch’s Method and the CSD approach, which can achieve a PBR rate of at least 60%. From this experiment we can see that there is no “one-size-fits-all” approach for PBR in wireless networks with time-varying or unknown modulation types.

In Fig. 6(a), we further plot the PBR rate of the KLD approach with different transmission distances, considering the same modulation and packet size as in Fig. 5. It can be seen that the KLD approach can achieve a consistent PBR rate for most modulations. For example, a PBR rate of 82%, 80%, 82%, 82%, and 82% can be achieved for QPSK modulation at 50 samples per bin as the transmission distance increases from 0.5 meters to 2.5 meters. Similarly, almost a 100% PBR rate can be achieved at 100 samples per bin. The same trend can also be observed for other modulations, except 16-QAM. As shown in the right figure, the PBR rate of 16-QAM decreases from 92% to 50% as transmission distance increases from 0.5 meters to 2.5 meters. This is because high-order modulations can result in increased sensitivity to channel effects such as noise, fading, etc. We also notice that this sensitivity is accentuated with larger sample bins. As shown in the left-hand side plot, a more consistent PBR rate can be achieved for 16-QAM with smaller bin sizes. Therefore, smaller bin sizes should be used for high-order modulations at long transmission ranges for more consistent PBR performance.

In Fig. 6(b), we further study the PBR rate achievable with different packet sizes by considering 8-PSK and 16-QAM as example modulations for entropy and variance approaches. In general, a higher PBR rate can be achieved with 8-PSK modulation than with 16-QAM modulation in almost all the tested cases. For example, a PBR rate of 81% can be achieved for 8-PSK using the entropy method and packet size of 1500 bytes, which is 61% for 16QAM. This implies that in the context of protocol-agnostic sensing, higher-order modulation schemes can be used to reduce the probability of detection. It is also found from the left-hand side of the figure that for the entropy approach, a consistent PBR rate can be achieved for 16-QAM with different static packet sizes; in contrast, the right-hand side of the figure shows that a consistent PBR rate can be achieved for 8-PSK modulation using the variance approach. Due to the prevalence of packets smaller than 1500 bytes within each random-length set ($\sim 40\%$), poor performance on traffic with randomized packet length is expected given the performance degradation observed with smaller packet sizes for most observed methods. From these results, it is further verified that no single statistical approach to PBR will perform well across all the tested variables.

PBR Accuracy Analysis. In this experiment, we study the PBR accuracy performance, taking KLD and variance approaches as an example. As discussed in the beginning of this section, PBR accuracy is defined as the relative difference between the predicted and ground truth packet boundaries. This accuracy measure is determined by comparing the ground truth

boundaries to each predicted boundary, and is defined as $\Delta G_{prd} = \frac{1}{G_{prd}} \sum_{n=1}^{G_{prd}} |g_{bin(n)} - g_{bin,prd}(n)|$, where G_{prd} is the predicted number of packets in a sample stream, $g_{bin}(n)$ is the ground truth bin location of the n -th boundary, and $g_{bin,prd}(n)$ is the corresponding predicted bin location.

The results are reported in Fig. 6(c), where QPSK and GMSK modulations are considered using KLD and variance methods for packet sizes of 2000 and 4000 bytes. It can be seen from the left-hand side figure that the KLD-based method accurately predicts QPSK packets, while the achievable prediction accuracy is significantly lower for GMSK-modulated packets. For example, we observe a ΔG_{prd} of 5.04 bins (each of 50 samples) for the KLD method on QPSK-modulated packets of size 2000 bytes, which is 71.98 bins for GMSK-modulated packets. Similar results can be observed in the right-hand side of Fig. 6(c) for the variance approach. The sampling rate was configured in the receiver USRP to be 1,000,000 samples per second. Therefore, an offset of 5.04 bins corresponds to $252 \mu s$, and 3.6 ms for 71.98 bins offset. While these ground truths are not attainable in a protocol-agnostic context, this is an important metric by which to validate PBR algorithms. This implies that existing statistical approaches are more suitable for PBR detection in low data rate wireless systems with long packets, for example LoRaWAN and LTE networks.

3.2 OSWireless Development

3.2.1 Motivation

Software-defined networking (SDN) is a key technique to enable next-generation (NextG) *programmable* networks with high scalability and low management complexity, and to accelerate the adoption of new communication techniques and hence hasten the network evolution [27–30]. In the past decade, SDN has captured extensive attention of both the academia and industry [31–42]. Notably, in 2011 the Open Networking Foundation (ONF) released OpenFlow, a standard defining the communication interface between the control and data planes of SDN-based networks [27]. Later, Google deployed their first SDN-enabled backbone wide area network called B4 based on OpenFlow for connecting their worldwide data centers [40]. Recently, Google deployed a new-generation distributed SDN controller called Orion in Google’s B4 networks and data center Jupiter [42].

While the immense performance gain of SDN has been successfully demonstrated in wired networks (e.g., data center, backbone networks), the extension of SDN to wireless networks is however very challenging. The primary reason is that traditional SDN typically requires reliable high-data-rate links connecting the distributed forwarding substrates to the remote centralized SDN controller, which are typically not available in wireless networks. To address this challenge, significant efforts have been made to push the control plane closer to the edge of wireless networks. Representative examples include software defined radio access network (SoftRAN) [39], Software-defined cellular networks (CellSDN) [35], and mobile cloud computing based software-defined wireless network (MCC-SDWN) [43]. However, these approaches have been focusing on *softwarization* of civilian cellular networks and still require

high-data-rate backhaul networks to accommodate the resulting overhead traffic, e.g., around 500 Mbps to support 50 micro cells each with a cell range of the order of one kilometer based on SoftRAN [39]. *This calls for new SDN architectures for NextG (both civilian and tactical) networks with densely deployed small cells, mobile even flying hotspots and heterogeneous network topology.*

In the meanwhile, existing SDN controllers focus primarily on providing services for *distribution abstraction* and *forwarding abstraction*, while very few efforts have been made for *specification abstraction*, where the objective is to provide simplified network models for mapping abstract specifications to physical operational configurations [44–46]. As a result, to configure manually the networks, the network operators need to simultaneously grapple with mathematical modeling, design of (possibly distributed) numerical algorithms, protocol design as well as the implementation of the resulting control logic in hardware. *This process is typically tedious and error prone.*

Contributions. In this work we try to address the above challenges by designing *OSWireless*, a new SDN controller that can enable self-optimizing software-defined NextG wireless networks. In a nutshell, *OSWireless* provides a control plane for optimizing software defined wireless networks with automated optimization program generation capabilities, by virtualizing four control plane functionalities, including intent, mathematical, algorithmic and forwarding specifications, and providing them as a service to the network operators. Based on *OSWireless*, network operators are allowed to specify in a centralized manner the control intent (i.e., *what to do*) using the specification abstraction APIs offered by *OSWireless*, while the control intent will be translated to (possibly distributed) operational control specifications (i.e., *how to do*) following a series of carefully designed steps.

As illustrated in Fig. 7, to enable automated specification translation, we design two planes for *OSWireless*, i.e., *Wireless Network Abstraction Specification (WiNAS) Plane* and *Optimization-as-a-Service (OaaS) Plane*. The former converts intent specifications defined using high-level APIs to the corresponding mathematical specifications, and the latter generates automatically operational optimization algorithms (i.e., algorithmic specification) that can be executed on edge nodes to determine their forwarding strategies at network run time. We showcase the automated specification translation capability of *OSWireless* by prototyping and deploying it over NeXT, a newly developed software-defined network emulation and experimentation system for NextG wireless networks, and defining a set of distributed cross layer network control problems. *To the best of our knowledge, OSWireless provides for the first time a practical, end-to-end solution towards zero-touch software-defined NextG wireless networks.* The main contributions of this work are summarized as follows.

- *Development of WiNAS Plane.* We first design the *WiNAS Plane*, which is the network abstraction engine of *OSWireless*. The objective of this plane is to provide the common language for the whole *OSWireless* controller to specify various network control intents and to establish the interfaces between the intent specification and the optimization engine *OaaS Plane*. To this end, four modules have been designed, i.e., *NetTopo*, *QoSPara*, *ParaModel* and

MathSpec, to integrate the major functionalities of the *WiNAS Plane* such as network element and topology definition, network modeling, and network control problem construction.

- *OaaS Plane Prototyping*. To showcase how to provide optimization as a service based on the language defined by the *WiNAS Plane*, we further prototype the network optimization engine of *OSWireless*, i.e., *OaaS Plane*. In the prototyping, we consider two families of widely adopted distributed network optimization approaches as an example: dual decomposition and successive convex optimization. To enable fully automated specification translation, the intent specifications defined using textual strings are first converted to the symbolic domain and then the coupling among the protocol layers and distributed nodes are extracted from the intent expressions based on the APIs of the *WiNAS Plane*.
- *OSWireless Deployment and Evaluation*. We demonstrate and evaluate *OSWireless* by

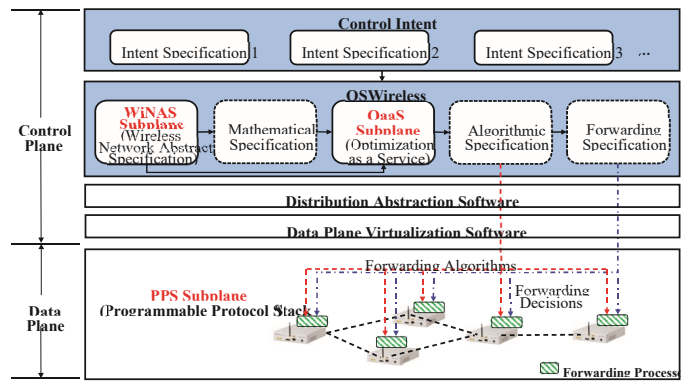


Figure 7: Architecture of *OSWireless* SDN Controller.

deploying it over NeXT, a newly developed network emulation and experimentation system. *OSWireless* is deployed over the edge server of the NeXT system to control two programmable networks, one implemented based on a universal broadband network mobile network emulator and the other based on USRP SDRs. We demonstrate the intent-driven automated optimization capability of *OSWireless* by considering a set of cross-layer network control problems. We also discuss the lessons learned through the development and demonstration of *OSWireless*.

The source code of *OSWireless* and *NeXT* will be released to the community via GitHub upon report publication.

3.2.2 Overall Objective and Design Approach

The primary goal of *OSWireless* is to enable intent-driven wireless networking with zero-touch optimization capability, by hiding the specification complexity from network controllers. Given a control intent (e.g., maximize the network spectral efficiency, minimize the end-to-end delay), *OSWireless* aims to determine in an automated manner the optimal or at least a desirable operational forwarding strategy (e.g., which nodes transmit with what power in which

frequency bands) that can be executed on the distributed physical forwarding infrastructure. Denote S_{int} and S_{fwd} as the intent specification and forwarding specification, respectively. Then the overall goal can be expressed as

$$f: C(S_{\text{int}}) \longrightarrow D(S_{\text{fwd}}), \quad (6)$$

where f denotes the functionalities provided by *OSWireless*, $C(\cdot)$ indicates that S_{int} is centrally defined, and $D(\cdot)$ represents that S_{fwd} are deployed on distributed physical forwarding substrates. Here, S_{fwd} and S_{int} represent the two extremes of network abstraction. By defining S_{fwd} directly, i.e., no or little abstraction, network operators are allowed to control all the forwarding details with the most flexibility, but at the cost of high specification complexity. Alternatively, different network applications can be developed independently to accomplish different control intents (i.e., S_{int}) with only the highest-level APIs exposed to network operators. This will hide most of the specification complexity while enabling only little reconfiguration flexibility. A natural question to ask is: how to achieve a good tradeoff between low complexity and high flexibility in defining the control specifications for optimizing software-defined wireless networks?

Design Approach. In this work we attempt answer this question by designing *OSWireless* following a three-phase approach. The objective of the first phase is to specify the control intent S_{int} using the APIs provided by *OSWireless* and in a centralized manner; and then construct the mathematical representation corresponding to intent specification S_{int} . Refer to the output of this phase as *Mathematical Specification* and denote it as S_{math} . In the second phase, *OSWireless* will generate a set of (possibly distributed) numerical solution algorithms to solve S_{math} . Denote the output of this phase as *Algorithmic Specification* S_{alg} . Finally, in the last phase S_{alg} will be executed at network run time to obtain the *Forwarding Specification* S_{fwd} for the distributed physical substrates. Then, the design goal of *OSWireless* in (6) can be rewritten as

$$f: C(S_{\text{int}}) \xrightarrow[\text{(i)}]{\text{Phase}} C(S_{\text{math}}) \xrightarrow[\text{(ii)}]{\text{Phase}} D(S_{\text{alg}}) \xrightarrow[\text{(iii)}]{\text{Phase}} D(S_{\text{fwd}}). \quad (7)$$

It is worth pointing out that in Phase 3 we consider distributed *Algorithmic Specification* S_{alg} as an example, while viewing the centralized implementation as a special case.

The rationale behind the four-phase design approach is to separate those coupled and manually handled network control processes, including network modeling, objective formulation and algorithm design, and provide the functionalities of each process as a service to the other processes. *This is similar to the TCP/IP protocol stack, which separates the network functionalities into five layers and provides the functionalities of each layer as a service to the other layers. In OSWireless, we focus on separating the functionalities for modeling and optimizing the programmable protocol stack of software-defined wireless networks.*

In this work, as illustrated in Fig. 7, we accomplish these three phases by designing two new sub-planes in the control plane, i.e., *Wireless Network Abstract Specification (WiNAS)*

Subplane and *Optimization-as-a-Service (OaaS) Subplane*, and a *Programmable Protocol Stack (PPS) Subplane* in the data plane.

3.2.3 WiNAS Subplane Design

The core functionalities of the *WiNAS Subplane* are two-fold: first, construct the corresponding mathematical specification S_{math} given a user-defined control intent specification S_{int} , i.e., Phase (i) in (7); and second, provide a common “*language*” for the other functionalities of *OSWireless*, including the *OaaS Subplane*, the mathematical, algorithmic and forwarding specifications. In *OSWireless*, these two functionalities are accomplished by further designing four modules for the *WiNAS Subplane*, i.e., *MathSpec Module*, *NetTopo Module*, *QoSPara Module* and *ParaModel Module*. The *MathSpec Module* is the place where the mathematical specification will be constructed, while the other three modules together provide the “*language*” service. Next, we first define the notations for simplifying the description of the four modules.

Notation Definition. In *OSWireless*, we refer to those network components and parameters that can be used to characterize the network behaviors as *Network Element*. There are two types of network elements: *View Element* and *Behavior Element*.

Definition 1 (View Element). *A view element is a network element that can be used to characterize the global view of the network. Here the global view refers to not only the traditional graph representation of the network topology based on vertices (i.e., nodes) and edges (i.e., links), but also the availability of the hardware and radio resources (e.g., antennas, subchannels) for each node and link. Denote the set of view elements as V .*

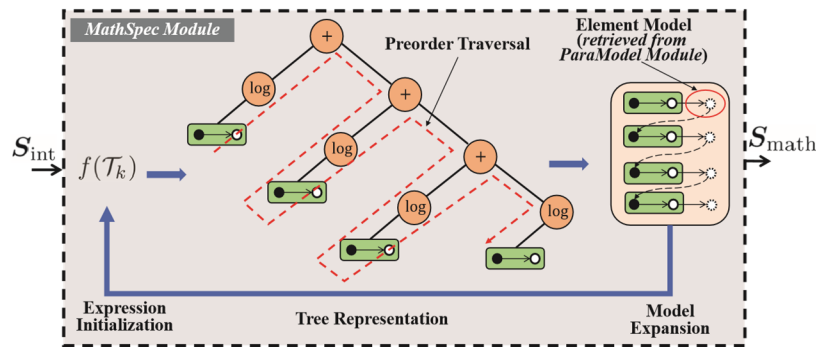


Figure 8: Diagram of MathSpec Module.

Definition 2 (Behavior Element). *A behavior element is a network element that can be used to characterize the behaviors of a view element in V in terms of various QoS metrics and based on other view and behavior elements, e.g., noise, capacity and delay. Denote the set of all*

behavior elements as B . Further denote the subset of behavior elements associated to network element $V_i \in V$ as $B(V_i)$.

A behavior element $B_i \in B$ is called *leaf behavior element* if it cannot be represented as a function of other behavior elements in B/B_i in the network control, e.g., noise, power; otherwise, it is called an *intermediate behavior element*, e.g., capacity, which is a function of noise and power. It is worth pointing out that, a behavior element can be either leaf or intermediate but cannot be both simultaneously in an intent specification S_{int} . The type of a specific behavior element depends on the adopted element model. For example, the capacity of a link will be viewed as a leaf element if it is considered as given or measured at network run time. The element models are defined in the *ParaModel Module*, which will be described later. Further define $T_i \triangleq (V_i, B_i)$ as a view-behavior element pair (VBEP) and denote the set of all VBEPs as T . Then, the intent specification S_{int} and mathematical specification S_{math} can then be represented based on a subset of VBEPs in T . The main difference between S_{int} and S_{math} is as follows. In S_{int} the network behaviors are characterized mostly based on intermediate behavior elements without exposing the lower-level details of the coupling among the behavior elements to the network operators. Differently, S_{math} characterizes network behaviors by formulating mathematically the corresponding network control problem by exposing all the details of the coupling among the involved behavior elements. The mapping between S_{int} and S_{math} is accomplished by the *MathSpec Module* as described below.

Mathematical Specification (MathSpec) Module. Given intent specification S_{int} , the *MathSpec Module* constructs the corresponding mathematical specification S_{math} following three steps: *Expression Initialization*, *Tree Representation* and *Model Expansion*, as illustrated in Fig. 8.

Algorithm 1: Tree representation of expression f

Data: The initial set of expressions $\mathcal{F}' = \emptyset$, $\mathcal{F} = \{f\}$, the initial VBEP set $\mathcal{T} = \emptyset$

Result: Updated set $\mathcal{T}$ of VBEPs contained in expression f

```

1 while  $\mathcal{F}' \neq \mathcal{F}$  do
2   Update  $\mathcal{F}'$ :  $\mathcal{F}' \leftarrow \mathcal{F}$ 
3   for Each expression  $g \in \mathcal{F}$  do
4     Update  $\mathcal{F}$ :  $\mathcal{F} \leftarrow \mathcal{F} \setminus \{g\}$ 
5     (Operator  $g_0$ , operands  $g_1, g_2$ ) = getCompExpr( $g$ )
6     Update  $\mathcal{F}$ :  $\mathcal{F} \leftarrow \mathcal{F} \cup \{g_1, g_2\}$ 
7     for Each operand  $g' \in \{g_1, g_2\}$  do
8       if checkVBEP( $g'$ ) is True then
9         Update VBEP set  $\mathcal{T}$ :  $\mathcal{T} \leftarrow \mathcal{T} \cup \{g'\}$ 
10      end
11    end
12  end
13 end

```

Roughly speaking, each intent specification S_{int} consists of a set K of textual expressions. Each expression defines either the utility or a constraint of the target network control problem, by

referencing to a subset of VBEPs using the APIs provided by the *NetTopo Module* (which will be described later in this section). Denote the set of VBEPs referenced in expression $k \in \mathcal{K}$ as $\mathcal{T}_k \triangleq \{(T_k^i)_{i=1}^{I_k}\}$ with $T_k^i \triangleq (V_k^i, B_k^i)$ and I_k being the number of VBEPs in T_k . Then the objective of this step is to construct an initial mathematical representation (denoted as $f(T_k)$) of the intent specification S_{int} . Figure 8 shows a toy example of $f(T_k)$ with four VBEPs. If we consider VBEP (link,rate) for each $T_k^i \in T_k$, where link and rate are respectively view and behavior elements, then the initial expression $f(T_k)$ can be represented as

$$\begin{aligned} f(T_k) &= \log(T_k^1) + \log(T_k^2) + \log(T_k^3) + \log(T_k^4) \\ &= \log(\text{link1.rate}) + \log(\text{link2.rate}) \\ &\quad + \log(\text{link3.rate}) + \log(\text{link4.rate}), \end{aligned} \tag{8}$$

i.e., sum-log-rate of the four links, a widely adopted utility function with log introducing proportional fairness among the links. In *OSWireless*, the expression initialization is achieved by simply replacing the textual API expressions with the corresponding VBEPs.

With the initial expression $f(T_k)$, the *MathSpec Plane* will then construct iteratively an expanded expression to characterize the mathematical coupling of the involved behavior elements. To this end, *MathSpec Plane* needs to identify the set of VBEPs contained in $f(T_k)$ in each iteration as the model expansion progresses. In *OSWireless*, this is accomplished by converting the textual expression $f(T_k)$ into a binary tree in each iteration. As illustrated in Fig. 8, each leaf node of the resulting tree is a VBEP and the other nodes are mathematical operators. To this end, preorder traversal is adopted to identify all VBEPs. For example, given the initial expression of $f(T_k)$ in (8), the *MathSpec Module* will construct in the first a binary tree with three nodes: operator “+”, leaf nodes $\log(T_1)$ and $\log(T_2)+\log(T_3)+\log(T_4)$. Since $\log(T_i)$ is not a VBEP, *MathSpec Module* further represents $\log(T_i)$ as a tree with operator log and leaf node T_1 (which is a VBEP). The tree representation stops until all the leaf nodes are VBEPs. The process of tree representation is summarized in Algorithm 1, where the job of $\text{getCompExpr}(\cdot)$ in line 5 is to retrieve the operator and operands contained in a textual expression and returns no operand if no operator detected in the expression; $\text{checkVBEP}(\cdot)$ in line 8 is used to determine if a textual operand is a VBEP. Both of these two functionalities are accomplished based on the “common language” services provided by the other three modules of the *WiNAS Subplane*, as described below.

Finally, each of the identified VBEPs is substituted with its corresponding model defined in the *ParaModel Module*. The resulting updated $f(T_k)$ will then be converted to a new binary tree. This procedure will be repeated until all the behavior elements (see Definition 2) in the expression are leaf behavior elements.

Common Language Service. Four types of “language” services are provided by the *WiNAS Subplane*. These are i) definitions of the view and behavior elements, ii) definitions of the associations of behavior elements to view elements, iii) modeling of the behavior elements, and

iv) operations of the view and behavior elements, such as $\text{getCompExpr}(\cdot)$ and $\text{checkVBEP}(\cdot)$ in Algorithm 1. These services are accomplished in three modules, i.e., *NetTopo Module*, *QoSPara Module* and *ParaModel Module*, as illustrated in Fig. 9.

The objective of the *NetTopo Module* is to enable flexible definition of different types of view and behavior elements, characterize the coupling among the defined elements and further associate them to the corresponding mathematical models. This plane also provides two types of control interfaces, i.e., internal interfaces for the different modules of the *WiNAS Plane* and external interfaces for intent specification and communicating with the *OaaS Plane*. To this end, the *NetTopo Module* manages a set of view records each corresponding to a view element in V referenced in intent specification S_{int} . Denoted the set of the records as

$$\mathcal{R}_v = \{R_v^n \mid n = 1, 2, \dots, N_v\}$$

where R_v^n represents the n th view record and N_v is the total number of the records. As illustrated in the top block of Fig. 9, each record R_v^n is a variable-length tuple, consisting of five categories of fields. These are $\text{vID}(R_v^n)$, which is the *OSWireless*-wide unique ID of view element R_v^n ; $\text{vAttrList}(R_v^n)$, which comprises the list of attributes defined for view element R_v^n , such as the name of the element (vElmtName), the parent view element that R_v^n is associated to (vPrntElmt), whether R_v^n is a set or an individual element (vElmtType), among others; field $\text{vNetPtr}(R_v^n)$ consists of a set of pointers, each pointing to another associated view element $R' \in \mathcal{V}_v$, e.g., the individual elements that forms a set element; Finally, $\text{vBPtr}(R_v^n)$ maintains the set of behavior elements attached to R_v^n and $\text{vOprPtr}(R_v^n)$ defines the set of the operations supported by R_v^n . It is worth mentioning that these operations have been designed to enable automated expression initialization and model expansion in the *MathSpec Module* (e.g., $\text{checkVBEP}(\cdot)$) and associate the behavior elements defined in the *QoSPara Module* to the corresponding behavior element (e.g., $\text{vBElmtLoader}(\cdot)$).

Similar to the *NetTopo Module*, the *QoSPara Module* maintains the set of behavior elements B . Similar to the view records, as illustrated in Fig. 9, each behavior record also consists of an *OSWireless*-wide unique ID $\text{bID}(R_b^n)$ and an attribute list $\text{bAttrList}(R_b^n)$. Examples of these behavior attributes include $\text{bLayer}(R_b^n)$, which provides the protocol layer information of the behavior elements; and $\text{bHID}(R_b^n)$, which defines the *node* view element that behavior element R_b^n is associated to. This information will be provided to the *OaaS Subplane* for automated distributed problem decomposition.

Finally, each behavior record $R_b^n \in B$ also has a bMdIPtr field, which is a pointer pointing to the mathematical model of the behavior element. All the models available to a behavior element are defined in the *ParaModel Module*. For each behavior element, two types of models have been defined in *OSWireless*: i) *Expression Model (EM)*, which models a parameter using closed-form mathematical expression; and ii) *Script Model (SM)*, which models a behavior element using a piece of script. Compared to expression models, script models can provide

more flexibility in characterizing the coupling among different behavior elements and hence are more suitable for defining sophisticated network control problems.

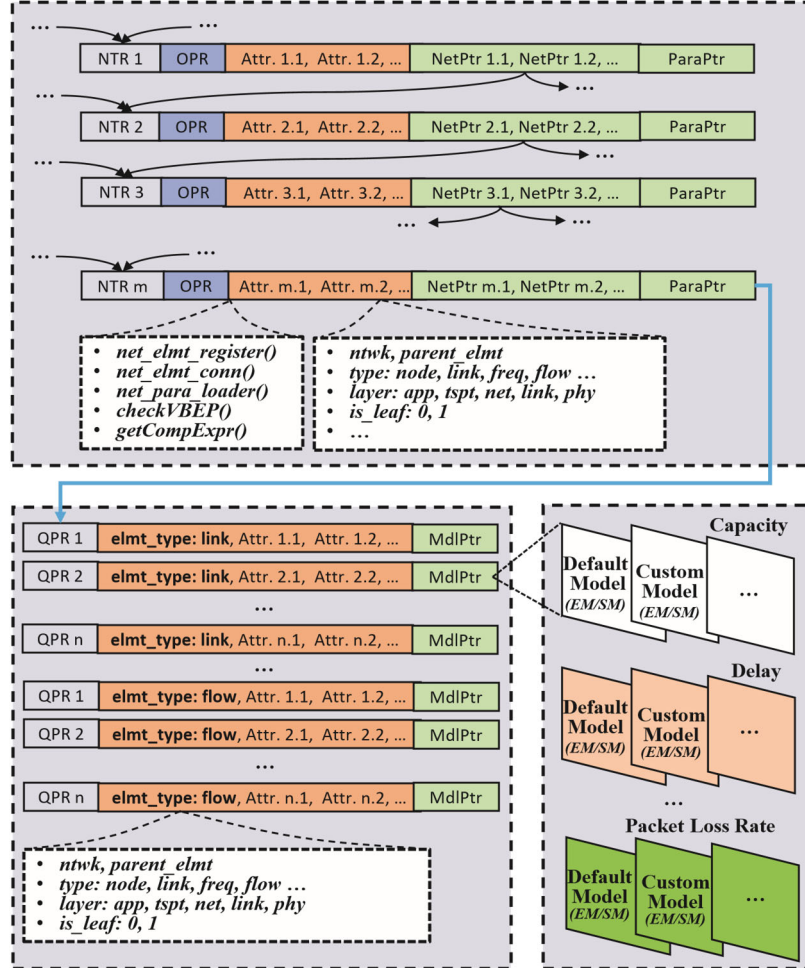


Figure 9: Diagram of NetTopo, QoSPara and ParaModel modules.

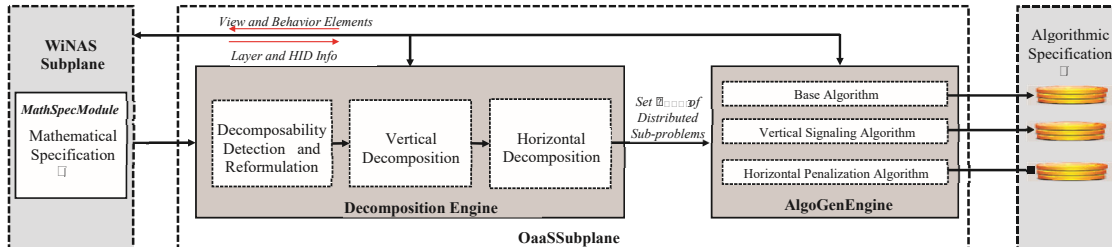


Figure 10: Overall architecture of OaaS Subplane.

3.2.4 OaaS Subplane Design

With the mathematical specification S_{math} obtained by *WiNAS Subplane*, *OaaS Subplane* will construct in an automated manner a set of numerical solution algorithms (i.e., the algorithmic specification as illustrated in Fig. 7) that can be executed on distributed forwarding substrates to solve S_{math} . Since the mathematical specification S_{math} is defined centrally and possibly across multiple protocol layers, we first need to decompose S_{math} into distributed ones. To this end, we design *OaaS Subplane* based on “layering as optimization decomposition” [47] and “decomposition by partial linearization” [48] theoretical frameworks and on the “common language” services provided by the *WiNAS Subplane*. It is worth pointing out that *OSWireless* is not limited to any specific decomposition and optimization theories. Our objective in this section is to showcase “optimization as a service” and the design can be easily extended to accommodate other theories, such as distributed optimization and statistical learning via the alternating direction method of multipliers [49], distributed asynchronous constrained stochastic optimization [50], and successive convex approximation for parallel and distributed optimization [51]. The architecture of *OaaS Subplane* is illustrated in Fig. 10, which consists of two major components, i.e., *Decomposition Engine* and *AlgoGen Engine*.

3.2.5 Decomposition Engine Design

Automated Decomposition Approach. Simply speaking, a given textual expression f can be decomposed in two steps: i) break down f into a set of component expressions connected with the operator “+”¹; and ii) assign the component expressions into different groups corresponding to different subproblems. The first step can be accomplished based on the tree representation service of *WiNAS Subplane*. Denote the resulting set of component expressions as $K_{\text{comp}}(f)$. In the second step, the component assignment is accomplished through two types of decomposition: vertical and horizontal decomposition, based on the layer and horizontal index (HID) information obtained from *WiNAS Subplane*, as illustrated in Fig. 10.

The objective of vertical decomposition is to uncouple the coupling among the protocol layers involved in mathematical specification S_{math} . To this end, for each component expression $k_{\text{comp}} \in K_{\text{comp}}(f)$ its associated layer is obtained using *WiNAS Subplane* API $\text{bLayer}(R_b(k_{\text{comp}}))$, where $R_b(k_{\text{comp}})$ represents the VBEPs contained in component expression k_{comp} . For example, for VBEP (ses, rate), i.e., the transport-layer transmission rate of a session, the corresponding component expression will be assigned to the transport-layer subproblem. Denote L as the set of subproblems obtained through vertical decomposition, with each subproblem involving one protocol layer.

¹ Operator “-” is considered as part of the component expression. For example, expression $x-y$ will reformulated as $x + (-1) * y$ before the tree representation.

Based on horizontal distribution, each subproblem in L is further decomposed into a set of subproblems each associated to a single node. To this end, we first introduce the notation of Horizontal Index (HID) in the following definition.

Definition 3 (Horizontal Index (HID)). *An HID is an identifier that can be used to identify the view network element where another behavior or view element should be registered and managed at network run time.*

Each behavior element is associated to a network view element as its HID when the behavior element is invoked for the first time during the construction of the mathematical specification S_{math} . By default, the HID will be the parent view element (i.e., parent `_elmt` in *QoSPara Module* of Fig. 9) of the behavior element. For example, the HID of behavior element link capacity will be its parent view element link; the HID of a link will be the source node of the link. In horizontal decomposition, to assign a component expression $k_{\text{comp}} \in K_{\text{comp}}(f)$ to a distributed problem, we only need to get its HID for using *WiNAS Subplane* API `bHID( $k_{\text{comp}}$ )` and further get the HID of `bHID( $k_{\text{comp}}$ )`. This process is repeated until the HID of the view element is itself and component expression k_{comp} will be assigned to the corresponding subproblem.

It is worth pointing out that in the above automated decomposition it has been assumed that expression f is decomposable. However, this is not always the case in wireless network modeling and optimization because of the coupled control variables of different protocol layers and different nodes. To address this challenge, a *decomposability Detection and Reformulation* module has been designed in the *OaaS Subplane*, as illustrated in Fig. 10.

Decomposability Detection and Reformulation. In *OSWireless*, we determine the decomposability of expression f by determining the decomposability of its component tree. As mentioned above, expression f can be represented as a tree with a set $K_{\text{sx}}(f)$ of component expressions connected with operator “+”. Then the decomposability of f depends on the decomposability of each branch of the tree. This can be simply accomplished based on the layer checking service provided by `bLayer( $R_b(k_{\text{comp}})$ )` and HID checking service by `bHID( $R_b(k_{\text{comp}})$ )`. Take vertical decomposition as an example. `bLayer( $\cdot$ )` first detects the behavior elements in k_{comp} and then retrieves all the corresponding protocol layers. If different layers are identified, it means the component expression is non-decomposable and cannot be assigned to a subproblem corresponding to any specific protocol layer. *How can we still decompose mathematical specification S_{math} in this case and in an automated manner?*

Notice that different decomposition theories have been established to tackle the decomposability problem within the “layering as optimization decomposition (LOD)” framework, such as dual decomposition, primal decomposition, hybrid decomposition and hierarchical decomposition as well as indirect decomposition [47]. *However, it is by no means easy to automate the decomposition based on these theories because they all require rather complicated analysis and reformulation of the specification expressions.* In this work, we aim to investigate this problem by prototyping for the first time *OaaS Subplane* considering indirect decomposition as an example while the design principles can also be extended to the automation of other decomposition theories.

Roughly speaking, a specification expression can be decomposed based on indirect decomposition by introducing auxiliary variables to i) uncouple the coupling among the variables in a non-decomposable expression and ii) coordinate the optimization of the resulting subproblems. Please refer to [47] for details of these decomposition theories. A toy example is the queuing delay model. If the M/M/1 model is considered, the average response time can be expressed as a function of the link capacity and the source rate, i.e., $1/(\text{link_capacity} - \text{src_rate})$. Here, the model is defined for $\text{link_capacity} > \text{src_rate}$, otherwise the average delay hence response time will become infinity. This expression is clearly not directly decomposable since link_capacity and src_rate operate at different (respectively physical and transport) protocol layers, and the expression will be viewed as a single branch in the tree representation. Based on indirect decomposition, an auxiliary variable v_{aux} will be introduced along with a decomposable equality constraint $v_{\text{aux}} == \text{link_capacity} - \text{src_rate}$. Another example of auxiliary variable is Lagrangian coefficient, which can be introduced in dual decomposition to absorb the constraints to utility function. Technically speaking, an auxiliary variable can be assigned arbitrarily to one of the involved protocol layers in the case of a tie. In *OSWireless*, the rules for auxiliary variable introduction and their assignment to protocol layers are defined in *Decomposition Detection and Reformulation Module* (i.e., this module) and in the behavior element models in *ParaModel Module* (Fig. 9). We omit the details of these rules because of the limitation of space.

3.2.6 AlgoGen Engine Design

Recall in (7) that the output of Phase 2 is the distributed algorithmic specification S_{alg} . To this end, *OaaS Subplane* further generates numerical solution algorithms to solve each of the subproblems obtained above. As shown in Fig. 10, three types of algorithms will be generated, namely *base optimization algorithm*, *vertical signaling algorithm* and *horizontal penalization algorithm*.

A set of numerical algorithm templates have been designed in *AlgoGen Engine* for the base algorithm construction. Each template defines the formats for the optimization variables, the signaling parameters and the penalization items. To this end, we define the algorithm templates based on CogApp, an open-source content generator for executing Python snippets in source files [52]. To construct the base algorithm, we first detect the optimization variables present in each of the distributed problems and substitute the detected variables into the algorithm template. The optimization variables are specified in intent specification S_{int} , and the corresponding information is stored in the behavior element in the *QoSPara Module* of *WiNAS Subplane* and can be retrieved using *WiNAS API* $\text{bVar}(\cdot)$ when constructing the base algorithm. Both scalar (i.e., $|\text{V}_{\text{opt}}| = 1$) and vector (i.e., $|\text{V}_{\text{opt}}| > 1$) variables can be supported by the *AlgoGen Engine*, where V_{opt} denotes the variables detected in the subproblem. The vertical signaling parameters can be detected and substituted into the algorithm template similarly. For example, if dual decomposition is considered, the vertical signal parameters will be those Lagrangian coefficients introduced when constructing the dual expression of the original expression. These parameters can be updated at network run time simply by a projected linear

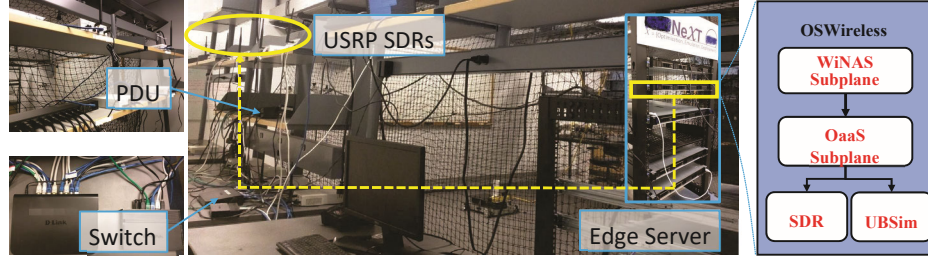


Figure 11: Architecture of the software-defined testbed.

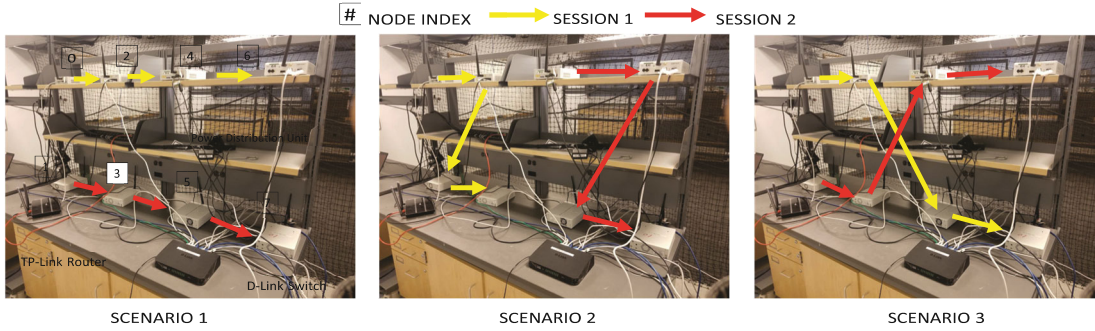


Figure 12: Network topology for OSWireless testing over NeXT testbed.

operation derived based on the corresponding constraint expression [53,54] and then exchanged among the subproblems where the parameters are involved.

The template algorithm has been designed to optimize a modified version of the original utility in each of the generated distributed subproblems. To this end, a penalization item is incorporated to the utility function to prevent a distributed node to hurt the others too much. In the current implementation of the *AlgoGen Engine*, we construct the penalization item considering the decomposition by partial linearization theory [48] as an example, while other penalization theories can also be incorporated in future work [51]. The same as in [55], the automated generation of the penalization items are accomplished by converting the textual expression into the symbolic domain [56], and we omit the details because of the limitation of space.

3.2.7 OSWireless Prototyping and Deployment

Recall that the *Algorithmic Specification* S_{alg} generated by the *OaaS Subplane* will be executed either in *OSWireless* in the case of centralized control or the distributed forwarding substrates in the data plane. In this work, we consider distributed control as an example while viewing the centralized control as a special case. To this end, we implement *OSWireless* and deploy it over NeXT, a newly designed software-defined testbed for wireless network modeling, optimization and deployment.

A snapshot of the testbed is shown in Fig. 11. There are two two major components in the NeXT testbed: edge server and front-end SDR. The edge server consists of five Dell EMC R340

powerededge workstations, each with Intel Xeon E-2246G 3.6 GHz CPU and Ubuntu v18.04. The front-end SDR consists of 20 N210 USRPs and is powered by three CyberPower PDU41001 Switched Power Distribution Units (PDU). The PDUs have been deployed to enable remote experimentation over the testbed during the COVID-19 pandemic and further share the testbed to the community². The edge servers and the front-end SDRs are connected using two switches with Gigabit Ethernet cables. *OSWireless* is deployed on one of the workstations as indicated by the yellow rectangular in Fig. 11, including *WiNAS Subplane* (Sec. 4.3) and *OaaS Subplane*. The other workstations serve as the controlling hosts of the front-end SDRs for baseband signal processing.

In addition to the software-defined data plane as discussed above, the *OaaS Subplane* is also interfaced with *UBSim*, as illustrated in Fig. 11. *UBSim* is a Python-based **Universal Broadband Simulator** for event-driven broadband integrated aerial-ground wireless networks. *UBSim* comprises of four modules: *Network Configuration Module (NCM)*, *Network Element Module (NEM)*, *Discrete Event Driver (DED)*, and *Algorithm Repository Module (ARM)*. The NCM module provides the APIs for configuring various virtual network elements, such as the number of nodes, frequency band, bandwidth; The NEM module defines the classes for all the network elements in a hierarchical manner. The DED module provides the discrete network simulation environment based on the open-source library SimPy [57]. Finally, the ARM module is the place where the algorithms automatically generated by the *OaaS Subplane* will be deployed and executed at network run time.

3.2.8 Experimental Evaluation

In this section, we aim to i) showcase how *OSWireless* can be used to hide the specification complexity by converting automatically a given control intent (*what to do*) to operational control programs (*how to do*); and ii) how effective are the automatically generated control programs at network run time. To this end, we conduct a series of experiments over the NeXT testbed considering network control problems at different protocol layers.

² Currently we are extending the NeXT testbed to enable experiments for integrated aerial-ground wireless networks and make the testbed publicly accessible by interfacing it with public cloud AWS.

```

1 # Network construction
2 ...
3
4 # Network behavior description
5 nt_ctl = ncp(nt)
6 expr = (expr_xpd, (expr_ori, ''))
7 allsess = nt.get_sess_set()
8 for sess in allsess:
9     x = nt.get_expr_new(sess, net_name.ssrates)
10    x = nt_ctl.mkexpr_new(x, 'log')
11    expr = nt_ctl.mkexpr_new(expr, x, '+')
12    expr = nt_ctl.mkexpr_new(expr, '(-1)', '*')
13    expr_name, info = nt_ctl.record_expr(expr)
14    nt_ctl.set_utilt(expr_name)
15    alllink = nt.get_link_set()
16    for link in alllink:
17        link_obj = nt.get_netelmt(link)
18        for link in alllink:
19            link_obj = nt.get_netelmt(link)
20            x = nt.get_expr_new(link, net_name.lkcap)
21            list_sess_name = nt.get_dependent_para(link, net_name.ses)
22            y = ''
23            for ses in list_sess_name:
24                z = nt.get_expr_new(ses, net_name.ssrates)
25                y = nt_ctl.mkexpr_new(y, z, '+')
26            expr = nt_ctl.mkexpr_new(y, x, '-')
27            expr_name2, info = nt_ctl.record_expr(expr, link)
28            nt_ctl.set_var(net_name.ssrates)
29            nt_ctl.set_var(net_name.lkpwr)
30
31 # Decomposition and algorithm generation
32 ...

```

Utility Specification

Constraint Specification

Variable Specification

Figure 13: Intent specification for *MSMH_Pow_ThrptRate_Max* based on *OSWireless* APIs written in Python 3.0.

Experiment 1: MSMH_PowRate_ThrptMax. In the first experiment, we consider networking scenarios with multi-hop device-to-device communications in NextG networks. Three scenarios are considered, as shown in Fig. 12. In each scenario, two source nodes communicate with their destination nodes via a set of relay nodes. The available spectrum is divided into three subchannels each of bandwidth 200 kHz, centered at 2 GHz, 2.2 GHz and 2.4 GHz, respectively. The control objective in this scenario is to maximize the sum end-to-end throughput of the two sessions by jointing controlling the transmission power of nodes at the physical layer and the source rate at the transport layer, subject to the proportional fairness between the two sessions. We refer to this scenario as *MSMH_Pow_ThrptRate_Max* where the first field (MSMH) represents a multisession multi-hop network topology, the second field indicates the control variables, and the last field is the control objective. The same naming convention will be adopted for the other scenarios for easier source code sharing in the future. Next we first showcase how to define the intent specification based on the APIs provided by *OSWireless*.

Figure 13 shows the main body of the intent specification for control problem *MSMH_Pow_ThrptRate_Max* written in Python 3.0. It can be seen that the network behavior description, including the specification of network utility, constraints and control variables, can be accomplished by referring to three types of behavior elements and their associated view elements using the high-level APIs provided by *OSWireless*. These behavior elements are *ssrate* (source rate of a session), *lkcap* (link capacity) and *lkpwr* (transmission power for a link), as indicated by the solid ellipses in Fig. 13. Notice that there is no need to specify explicitly the low-level coupling among these view elements, e.g., the mathematical model of *lkcap* as a function of *lkpwr*, their association to different protocol layers. Instead, these coupling will be integrated to the initial expression defined by the intent specification (i.e., *expr_ori*) through the expression expansion operation in the *MathSpec Module* (see Fig. 8). The resulting expanded mathematical specification is recorded in *expr_xpd* in Fig. 13.

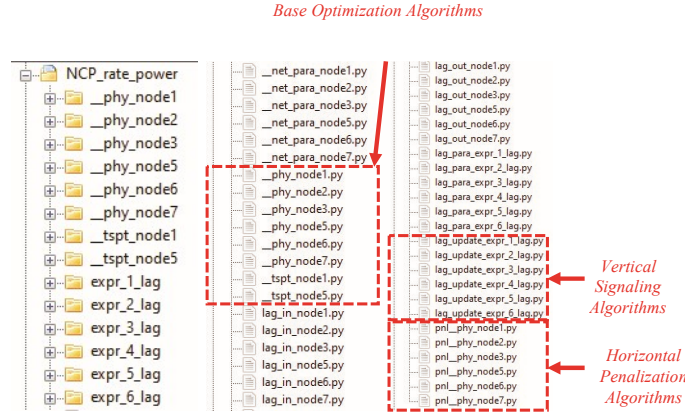


Figure 14: Program files of the distributed cross-layer algorithmic specification automatically generated by OSWireless for network control problem MSMH PowRate ThrptMax.

It is worth pointing out that the centralized high-level specification in Fig. 13 is all that is needed for OSWireless to generate the operational distributed cross-layer control programs for MSMH_Pow_ThrptRate_Max, while all the specification complexity with the following mathematical specification and algorithmic specification (confer to Fig. 7) will be taken care by OSWireless in an automated manner and hidden from the network controllers. The resulting algorithm specification is shown in Fig. 14 for each node at different protocol layers (physical and transport in this experiment), which consists of base optimization algorithms, Lagrangian updating algorithms for vertical signaling among protocol layers, and horizontal penalization algorithms. These operational programs will be loaded to the control host (i.e., the edge server in Fig. 11) and executed to control the distributed USRP SDRs at network run time.

Figure 15 shows the end-to-end throughput performance of the control programs automatically generated by OSWireless. The instantaneous throughput is reported in Fig. 15 (a) considering Scenario 1 (see Fig. 12) as an example. It can be seen that significant throughput gain (around 300%) can be achieved compared to the benchmark scheme (*Benchmark 1*), based on which the hardware transmit gain of the USRP SDRs is set to 15 dB (the middle of the range) and no rate or power adaptations are considered. Figure 16 reports the corresponding instantaneous throughput and transmission power. The average performance is plotted in Fig. 15 (b) for the three scenarios by averaging over 10 experiments. Three more benchmark schemes are considered in addition to that in Fig. 15 (a): *Benchmark 2*: maximum hardware transmit gain for the USRP SDRs (25 dB) and maximum source rate; *Benchmark 3*: fixed transmission power (15 dB) with adaptive source rate; and *Benchmark 4*: power adaptation subject to a target end-to-end throughput. It can be seen that OSWireless outperforms all these four benchmark schemes, which validates the effectiveness of automatically-generated control programs.

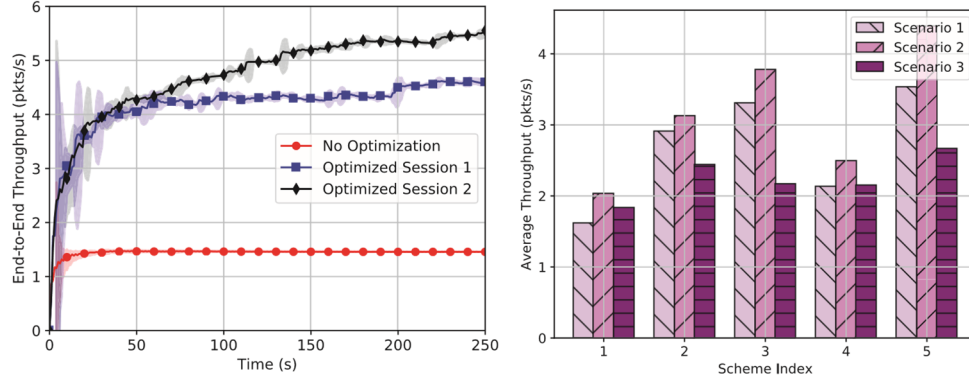


Figure 15: (a) Single-run running average and (b) multi-run average end-to-end throughput for *MSMH_Pow_ThrptRate_Max*

(a)

(b)

Experiment 2: *MSMH_PwrRate_DlyMin*. In the above experiment we have showcased through a toy example the operation of *OSWireless*, including intent specification, automated generation of the corresponding mathematical specification, as well as the effectiveness of the resulting control programs. In that example, the control variables *ssrate* and *lkpwr* are only loosely coupled through the network flow conservation constraint for each link (lines 18-27 in Fig. 13) and hence can be decoupled vertically after constructing the corresponding dual problem of the mathematical specification. In the following experiments, we further test the flexibility of *OSWireless* in control intent definition considering more sophisticated control problems.

In this experiment, referred to as *MSMH_PwrRate_DlyMin*, the control objective is to minimize the average end-to-end delay of concurrent sessions in a multi-session multi-hop network, by jointly controlling the source rate at the transport layer and transmission power of the nodes at the physical layer under the constraints of minimum end-to-end throughput for each session. The M/M/1 queuing model (see [58] for detailed expressions of the model) is considered for each session. In this case, the session rate *ssrate* and link capacity *lkap* (hence the transmission power *lkpwr*) are closely coupled in the utility function through the queuing model where they appear in the numerator of the model at the same time $1/lkap(lkpwr) \cdot ssrate$, and the resulting mathematical specification cannot be decomposed directly and more sophisticated decomposition techniques are needed.

Fortunately, *OSWireless* can hide most of the specification complexity. Specifically, we just need to define the control intent by simply calling the following model installation API: `nt.install_model(sess, net_name.ssdelay, net_name.mm1)`, where *sess*, *ssdelay* and *mm1* are respectively the view element, behavior element and the parameter model defined in the *NetTopo*, *QoSPara* and *ParaModel* modules of the *WiNAS Subplane*, as illustrated in Fig. 9. Then, the resulting mathematical specification will be analyzed for decomposability detection, reformulated by introducing auxiliary vertical signaling variables to upcouple *lkpwr* and *ssrate* in the queuing model, and finally decompose the obtained network control problem based on indirect decomposition. *Notice that all of these operations except the intent specification are conducted automatically by OSWireless against manually based on traditional network control.*

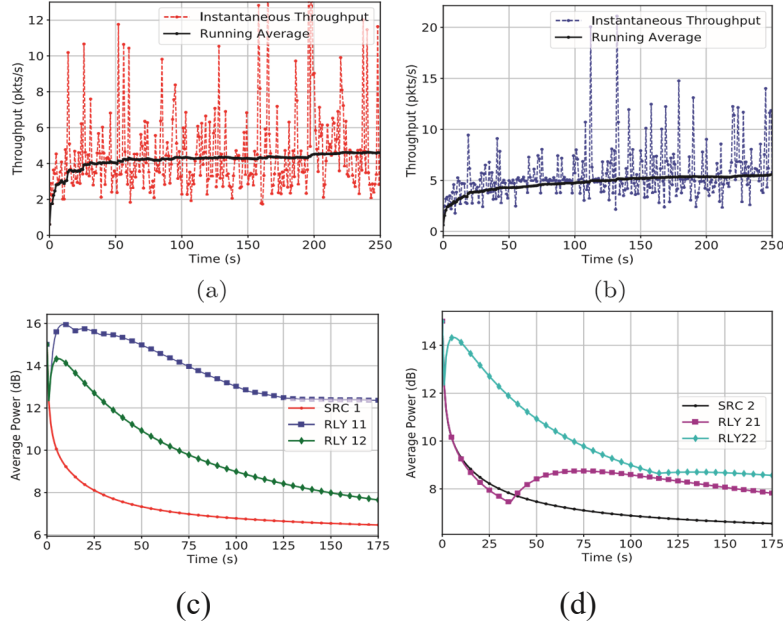


Figure 16: Instantaneous throughput of (a) Session 1 and (b) Session 2; (c), (d): Instantaneous transmission power of network control problem MSMH_PowRate_ThrptMax in Scenario 1.

The delay performance of the obtained algorithmic specification is reported in Fig. 17. In the experiment, we consider low-data-rate latency-critical applications, with the target end-to-end throughput set to 1.5 kbps and 3 kbps for the two sessions, respectively. From Figs. 17 (a) it can be seen that the end-to-end delay can be effectively reduced for both of the two sessions. The corresponding end-to-end delay is plotted in Fig. 17 (b). We can see that the target end-to-end throughput can be assured for them both as the delay is minimized. In Fig. 17 (b) we compare *OSWireless* to three benchmark schemes: benchmarks 1, 3 and 4 in Experiment 1 but tailored for this experiment. The results are obtained by averaging over 10 experiments. We can see that an average delay of 0.14 s can be achieved by *OSWireless*, which outperforms all the three benchmarks (0.86, 0.54 and 0.41 seconds lower, respectively). *This experiment further verifies the correctness of the control programs automatically generated by OSWireless and hence the capability of OSWireless in hiding the specification complexity for sophisticated network control intents.*

Experiment 3: MSMHMob_Mov_ThrptMax. In this experiment, we further test *OSWireless* considering mobile scenarios based on UBSim (confer to Fig. 11). Three scenarios are considered as shown in Fig. 18(a): 4) *symmetrical network*, 5) *asymmetrical network* and 6) *special case of Scenario 4*.³ In each scenario, the locations of the nodes are randomly initialized. The control objective is to maximize the end-to-end throughput of concurrent sessions by

³ The network scenario index starts from iv to avoid confusion with the three scenarios in Fig. 12.

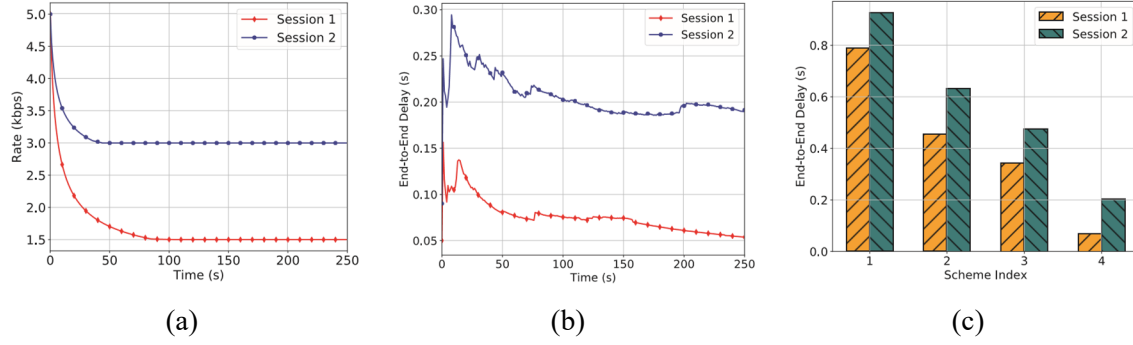


Figure 17: Instantaneous (a) transmission rate and (b) end-to-end delay; (c) average end-to-end delay.

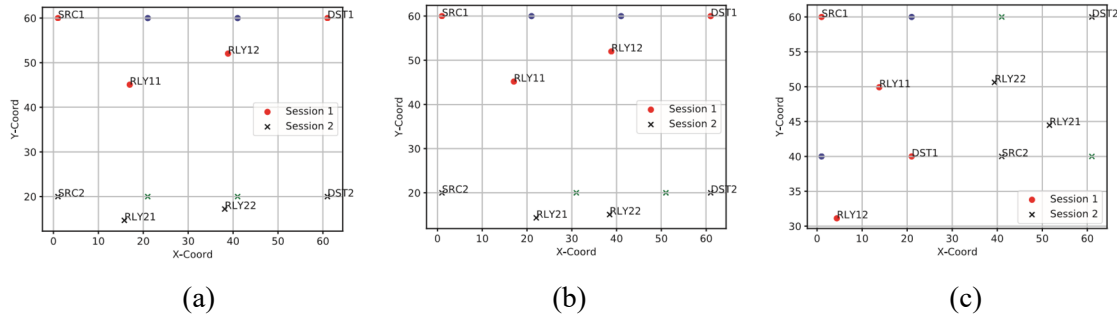


Figure 18: Optimized location for (a) Scenario 4, (b) Scenario 5 and (c) Scenario 6.

jointly controlling the source rate and the movement of the relay nodes. Notice that the locations of the source and destination nodes are assumed to be determined by the certain tasks and fixed during the experiments. We refer to this experiment as *MSMHMob_Mov_ThrptMax*.

Based on *OSWireless*, the control intent in this experiment can be defined similarly as in Fig. 13. The only major modification is to specify node coordinates `coord_x` and `coord_y` as control variables. This can be done by simply using two lines of code: `nt_ctl.set_var(net_name.coord_x)` and `nt_ctl.set_var(net_name.coord_y)`. Figure 18(a) reports the simulation results for Scenario 1, where the initial locations of the relay nodes of the session 1 is represented with blue solid circle, and green 'x' for session 2. The corresponding end-to-end throughput is shown in Fig. 19 (a). We can see that both the sessions are able to achieve the maximum achievable end-to-end throughput of 200 kbps in around the 200th time slot. More results are available in Figs. 18 (b)(c) and Figs. 19(b)(c). It can be seen that the end-to-end throughput can be effectively improved in all the test cases based on automatically generated control programs, which further verifies the flexibility and effectiveness of *OSWireless*.

3.2.9 Discussions

In this work, we provided a prototype design of *OSWireless* which automatically abstracts the high-level network control specification defined by the user and generate distributed

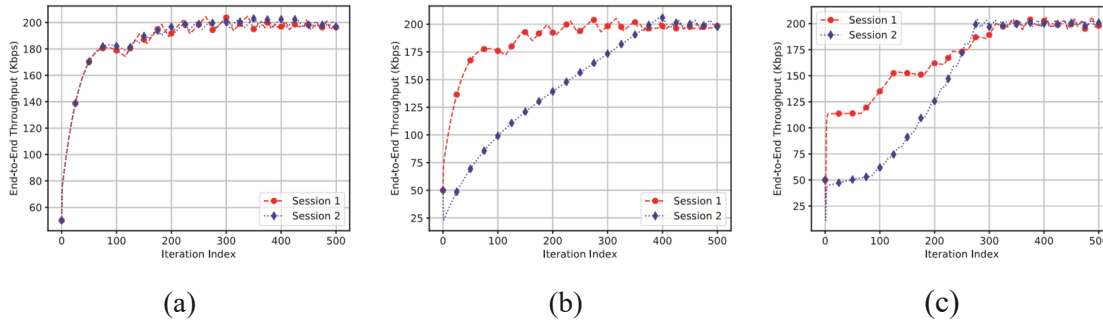


Figure 19: End-to-end throughput for (a) Scenario 4, (b) Scenario 5 and (c) Scenario 6.

cross-layer solution algorithm that can be implemented on either a SDR-based testbed or simulation platform. We envision that our *OSWireless* can significantly reduce the burden of the researchers by automating all the lower-level details. This prototype design has various avenues of improvement which will be considered in the next generation design of *OSWireless*. Some of the key features to be implemented are outlined below.

Standardization of Interfaces. In the current design of *OSWireless* it requires the users to modify the main core of *OSWireless* in order to add a new network element or attribute, implement a new decomposition approach, etc. In the next generation of *OSWireless* we intend to provide a wide array of APIs that would allow the users to easily add new elements or decomposition technique without modifying the main core. Additionally, the next-generation would also allow the users to define new control problem such as energy-minimization, data-freshness as well as complex control problem that includes various frequency bands such as Wifi/LTE, access techniques such as OFDMA/CSMA, MIMO beamforming to name a few.

Machine Learning Integration. Given a network element attribute, the models are pre-defined which use a closed-form expression to characterize the attribute. The next generation of *OSWireless* is expected to support machine learning techniques such as Echo State Learning, Reinforcement Learning, among others to characterize the attribute model. These machine learning models will prove valuable in case of a complex network control problem with various parameters that may affect the network and cannot be accurately modelled using a closed form expression.

Remote Access and Control. In order to test new algorithms developed based on our *OSWireless*, a one-of-a-kind cloud-based remote access and control framework is under development and will be made accessible to the research community which would allow researchers to conduct remote experiments without having to develop their own testbed. The key feature of this remote access framework is that it will allow the researchers to remotely

access and control the testbed as well as visualize the testbed and movement of mobile drones with the help of a “*all-in-one*” web-based GUI.

3.2.10 Related Work

There are a few existing SDN architectures for wireless networks, including (SoftRAN) [39], software-defined cellular networks (CellSDN) [35], mobile cloud computing based software defined wireless network (MCC-SDWN) [43], and their variations. For example, in software defined radio access network (SoftRAN) [39], to alleviate the traffic load pressure of the backhaul network, the centralized controller of SoftRAN makes only those decisions that have network-level influence, such as handover and transmission power control, while pushing latency-sensitive local decisions, e.g., downlink resource block allocation, into remote radio head controllers. Software-defined cellular networks (CellSDN) [35], mobile cloud computing based software defined wireless network (MCC-SDWN) [43] share the similar *split-designcontrol-plane* principles. Recently O-RAN has emerged to support interoperation between vendors’ equipment by providing industry-wide standards for RAN interfaces and to enhance the RAN performance through virtualized network elements and integrated intelligence in RAN [59,60]. Readers are referred to [61] for an excellent survey of the main results in this field. Different from these works, which primarily focus on softwarization of cellular networks, in this work we focus on automated generation of distributed optimization programs in NextG wireless networks.

Network automation has also attracted significant research attention [62–72]. For example, in [62] the authors develop a human-intervention-in-the-loop quasi-automated model calibration scheme for power budget control and site selection in cellular networks. In [67] Harte et al. design a tool called “THAWS” to speed-up the development and deployment of heterogeneous WSNs by allowing users to select different network parameters such as sampling frequency and network topology and then generating the corresponding code automatically. The authors of [68] design a toolflow that can help monitor the correctness of the implementation throughout the development of wireless sensor networks (WSN). In [71] Shimizu et al. present a modeling language that can simplify the WSN prototyping and optimization by allowing developers to define the network and models in a flexible and descriptive way. While these works can certainly simplify the network development, they still require significant human intervention in field measurements or model description.

The work most related to ours is the open-source project WNOS [55,73], where Guan et al. design an optimization-based wireless network operating system for principled software defined wireless networking [55,73]. WNOS shares the same design goal with *OSWireless*, i.e., providing cross-layer distribution optimization as a service in NextG networks. However, the major limitation with WNOS is that the automated decomposition of centralized network control programs is enabled by a technique called Disciplined Instantiation (DI), which can support only limited set of network control applications [74,75] because of the lack of flexibility in the instantiation of abstract network control problems. In *OSWireless*, we aim to

provide a new control plane that is not based on DI and hence can support more sophisticated and practical network control applications.

3.3 CloudRAFT: A Cloud-based Framework for Remote Experimentation for Mobile Networks

3.3.1 Motivation

Over the past decade, significant efforts have been made by the wireless community to build open, shared experimental facilities. A notable effort is the National Science Foundation (NSF) Platforms for Advanced Wireless Research (PAWR) Program [76]. The goal of PAWR is to develop four city-scale shared experimentation platforms for advanced wireless research. As of today, three PAWR platforms have been developed: the Platform for Open Wireless Data-driven Experimental Research (POWDER) [77], the Cloud enhanced Open Software defined Mobile wireless testbed for city-scale deployment (COSMOS) [78], and the Aerial Experimentation Research Platform for Advanced Wireless (AERPAW) [79]. A fourth platform, the Wireless Living Lab for Smart and Connected Rural Communities (aka ARA), was launched recently [80].

While these existing facilities have significantly advanced the experimental research for next-generation wireless systems, it is still challenging to fully meet the needs of experimental wireless research in terms of *diversity*, *generalizability* and *accessibility*. First, existing community experimentation platforms focus on rather specific areas in wireless research (e.g., AERPAW focuses on wireless UAV systems), while there are many research areas in wireless systems that still lack well-designed community shared experimental facilities such as sensing and networking for robotic autonomy, mobility intelligence, underwater and underground communications and networking, among others. Second, existing facilities are deployed in pre-selected environments with nodes installed at fixed locations. For example, PAWR platforms focus on experiments “in-the-wild” with wireless systems deployed in outdoor environments, however it is challenging to test the generalizability of the obtained results to different environments (e.g., indoors) or climates (which may affect significantly the signal propagation behavior, particularly in mmWave and terahertz frequency bands [81,82]). The same concern also applies to experimental research in data-driven wireless systems, where it is crucial to assure the generalizability (or transferability) of the trained models and the resulting artificial intelligence (AI) and machine learning (ML) algorithms [83]. Moreover, as experimental research becomes more and more important for the wireless community, the needs of experimental resources (e.g., software-defined radios, computing capability, and bandwidth) may exceed what can be offered by existing community facilities in the near future. *These challenges cannot be addressed without a systematic shift in the paradigm of sharing experimentation facilities for mobile networks.*

In this report, we propose a new approach called CloudRAFT for remote experimentation and testbed sharing for wireless mobile networks. At the core of CloudRAFT is to develop an open control plane based on a public cloud for management of the data, code and hardware resources associated with the testbeds. We claim the following two contributions in this report.

- We first propose the CloudRAFT framework, based on which experimenters are allowed to remotely access and control experimental resources via public cloud Amazon Web Service

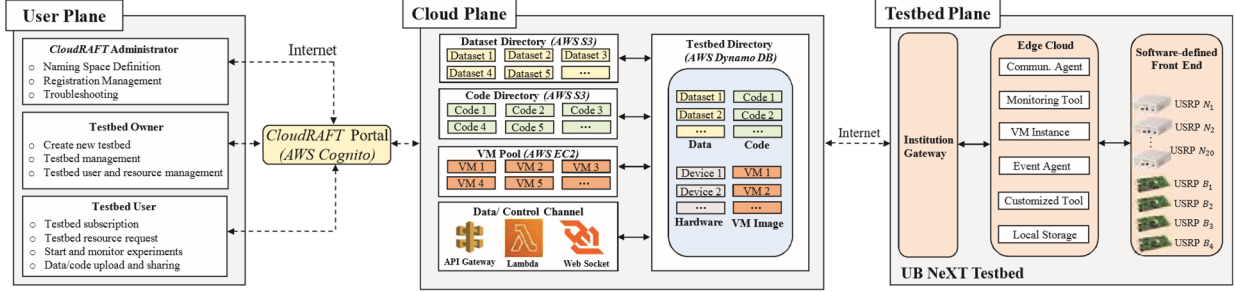


Figure 20: CloudRAFT: Cloud-based Remote Access for Wireless Testbeds.

(AWS) [84] and share the resulting data and code via the cloud. The enabling techniques are also discussed, including Amazon serverless service, Virtual Network Computing (VNC)-based remote command line, and Websocket-based real-time communications.

- We test these enabling techniques by connecting UB NeXT to AWS. UB NeXT is a software-defined testbed that has been developed at the University at Buffalo for wireless network modeling, optimization and deployment. Demonstrations of remote experiments over UB NeXT via AWS are also presented.

3.3.2 CloudRAFT Framework

The overall architecture of CloudRAFT is illustrated in Fig. 20, which consists of three planes: *User Plane*, *Cloud Plane*, and *Testbed Plane*.

User Plane. CloudRAFT uses this plane to manage three types of users, i.e., the *CloudRAFT Administrator*, the *Testbed Owner*, and *Testbed User*. The CloudRAFT administrator will oversee the operation of CloudRAFT, including user and testbed registration management, name space management for testbed resources, and troubleshooting. Testbed owners can create new profiles for their testbeds, manage the testbed profiles by adding testbed devices (e.g., USRP software-defined radio (SDR) [85]) and configure their parameters (e.g., the USRP type and IP address), approve and manage testbed users, and schedule and oversee the experiments on their testbeds. Lastly, testbed users will be allowed to subscribe to those testbeds connected to CloudRAFT, reserve testbed resources (e.g., computing and SDR resources), conduct experiments and monitor their status and output, and finally share the collected datasets via CloudRAFT. All the users will access CloudRAFT via the Internet using a CloudRAFT web portal deployed on AWS. User authorization and access control for CloudRAFT resources, including the dataset repository, code repository and virtual machine (VM) pool will be powered by AWS Cognito, a user identity, data synchronization and secure management service provided by AWS [86]. In this report, we design

CloudRAFT based on AWS taking the advantages of AWS's well-developed integrated web hosting, data storage and cloud computing services.

Cloud Plane. The primary challenges in testbed sharing are at least two-fold. First, it is complicated and costly for testbed owners to develop and deploy their own dedicated software, hardware and user interfaces for remote access, user authorization, experiment scheduling, and data storage. Second, for researchers, the complexity, inflexibility, and non-uniformity in accessing different testbeds can discourage their use. For example, most testbeds are deployed in subnets and can only be accessed through the corresponding local gateways. To access a testbed, a user needs to contact the testbed owner first, who will then contact the gateway manager to add the user to the allowed-user list. This whole process is time-consuming and repels interested users.

To address these challenges, as illustrated in Fig. 20, the Cloud Plane provides software and tools that can be used for testbed owners to conveniently share their testbeds with the community; hide the complexity in accessing different testbeds by enabling researchers to register a single CloudRAFT account and then subscribe to different testbeds; provides a bridge between researchers and testbed developers; and finally accelerates the formation of a mature ecosystem for experimental research for the wireless community focusing on mobile networks. In CloudRAFT, this can be accomplished by five major modules of the Cloud Plane, i.e., *Testbed Directory*, *Dataset Repository*, *Code Repository*, *Virtual Machine (VM) Pool*, and *Data and Control Channel*.

Testbed Directory. This directory maintains the basic information of the testbeds connected to CloudRAFT and hence increases their visibility to users. For each testbed, a table will be created and managed based on DynamoDB, a fully managed proprietary NoSQL database service provided by AWS [84]. Each table includes general information, such as the testbed name, testbed owner, supported research topics, allowed-user list, among others; and resource information, such as the hardware devices available to the testbed and their parameters, e.g., the type, image version and IP address of the USRP SDRs, the list of datasets and software that are applicable to the testbed, the list of VM images prebuilt for the testbed, and others. Each table also tracks the corresponding testbed's usage status, the availability of its devices (e.g., online, offline, and occupied) as well as upcoming scheduled experiments. The design of the Testbed Directory is extendable i.e., once a new testbed is created in CloudRAFT by its owner, a blank testbed profile and the associated DynamoDB tables will be created and initialized automatically, which can be further configured by the testbed owner.

Dataset and Code Repositories. In order to store and manage the experimental datasets and code, a dataset and code repository will be designed and hosted on AWS S3 (Simple Storage Service), which provides object storage through a web-service interface. Each dataset (or code) will be associated, through the above-discussed testbed directory tables, to one or more testbeds so it will be accessible to the corresponding testbed users. Once an experiment begins, the user will be able to load the selected dataset (or code) either to the VM launched in the *VM Pool* within

the *Cloud Plane* through a high-speed connection (25 Gbps), or to the edge cloud in the *Testbed Plane* using a data channel established based on “AWS SDK for Java” (which provides a Java API for AWS infrastructure services including AWS S3, EC2 and DynamoDB). New datasets collected during experiments will be first uploaded to the dataset repository, and later loaded to the VM for annotation and sharing. The *Dataset and Code Repositories* will effectively alleviate the burden of data/code storage, management and sharing for both testbed owners and users.

VM Pool. For each experiment, one or more VMs will be launched using the AWS EC2 (i.e., Elastic Compute Cloud) service. For each testbed, CloudRAFT will provide a set of prebuilt VM images with pre-installed software and libraries as needed for the experiments. Through the launched VM, the experimenter can i) execute *non-time-sensitive* programs in the VM, e.g., routing, transport-layer rate control, motion control, among others; ii) access the reserved edge cloud using ssh (or frp if the edge cloud is deployed in a subnet) to execute *time-sensitive* programs, e.g., baseband signal processing, scheduling, and power control; and iii) monitor the status and output of the experiments. The VM pool will be interfaced with the dataset/code repository via high-speed in-datacenter connections.

Data and Control Channel. Two types of channels will be established for each experiment, a data channel and a control channel, to bridge the Cloud Plane and the Testbed Plane. The data channel transfers datasets and code from the Dataset/Code Repository to the testbed edge cloud, and vice versa. The control channel sends control commands from the Cloud Plane to the testbed. The testbed will then execute the commands on the edge cloud. For example, by sending the command `uhd find devices` to the edge cloud (which is connected to front-end SDRs via Ethernet cables), the edge cloud will return information about the currently active and available USRP SDRs. The two channels are designed based on a combination of *AWS Lambda*, *Gateway API* and *WebSocket API* services. AWS Lambda acts as the serverless computing service, runs code in response to events and automatically manages the computing resources required by that code, and *Gateway API* and *WebSocket API* act as the “front door” for applications to access data, business logic, or functionality from backend services, e.g., the predefined Lambda functions.

Testbed Plane. The Testbed Plane provides a set of three tools in addition to the custom tools made available by the testbed owners. These include the *Communication Agent*, which serves as a relay between the Cloud Plane and the edge cloud for data transfer; the *Event Agent*, which receives and executes control commands from the cloud plane; and the *Monitoring Tool*, which feeds the real-time status of various computing and SDR devices to the Cloud Plane.

3.3.3 Enabling Techniques

In this section we identify several enabling techniques for CloudRAFT, including Amazon’s serverless service (ASS), websocket based real-time communications, VNC-based remote command line, and fast reverse proxy.

Amazon Serverless Service. With ASS [87,88], the cloud service providers take care of infrastructure management tasks allowing testbed and CloudRAFT developers to focus on the program development. This is very beneficial in the design of the cloud plane of CloudRAFT. The ASS offers a wide array of Application Programming interfaces (APIs) including REpresentational State Transfer (REST) API Gateway, Websocket API Gateway, Lambda Function, DynamoDB, Simple Storage Service (S3), and Cognito, among others.

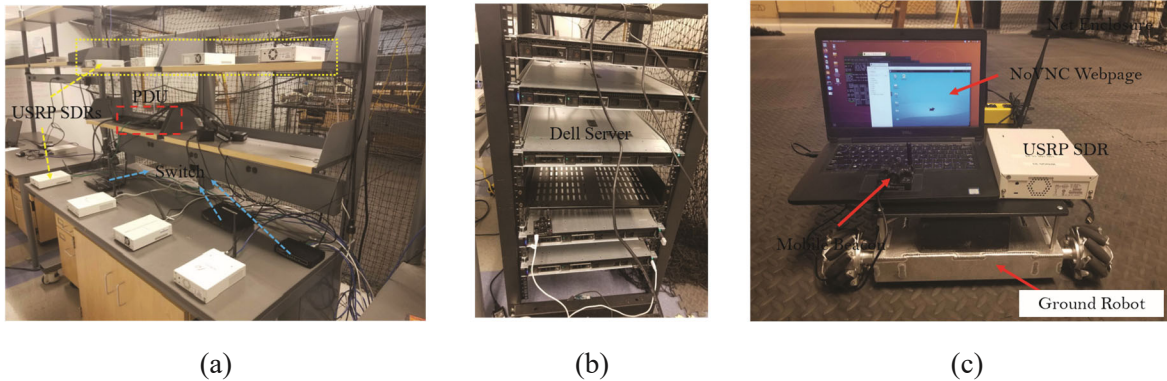


Figure 21: (a) Snapshot of USRP SDR deployment, (b) edge servers, and (c) ground robot vehicle.

The REST and Websocket API Gateways act as a firewall that will block, redirect or forward the requests from the users. They also take care of detection and execution of specific parameters in the user requests. The Lambda Function takes care of the execution of custom code defined by the testbed developers based on the parameters in the user requests. The DynamoDB is a key-value and document database that delivers low-latency responses. S3 is a static file storage server and provides easy-to-use management features and also configures the access controls based on the experiment requirements. Finally, Cognito provides a simple user-friendly GUI that allows users to sign up, log in and control their access to CloudRAFT via a webportal.

Websocket-Based Real-Time Communication. The Websocket-based real-time communication is the backbone of the CloudRAFT framework that takes care of the communications between the CloudRAFT control plane and the testbed edge server. These functions can be designed based on the WebSocket protocol. When the user types a command in the webpage, the command will be sent to an AWS Websocket API Gateway, which will then forward the command to the related Lambda function according to the “action” parameter in the WebSocket requests. The Lambda function will then further forward the command to the edge cloud servers. A Python script will run in the edge server waiting for the messages from the Websocket Gateway. Upon reception of the commands, the script will execute them and return the results to the WebSocket API Gateway, where the results will be returned to the experimenters. The same logic is applicable for the real-time update of testbed resources except

that the periodic signaling messages will be transmitted between the edge cloud server and the AWS cloud.

The WebSocket API gateway manages a routing table with different actions and the corresponding call functions. For example, the routing table consists of two special route keys, “connect” and “disconnect”. The former is used when a user wants to establish a connection to the edge server and the latter will be used for tearing down the connection. The server maintains a data table that contains all active hosts connected to the server. When the client host sends the connect message with action: \$connect, this request will be forwarded to the

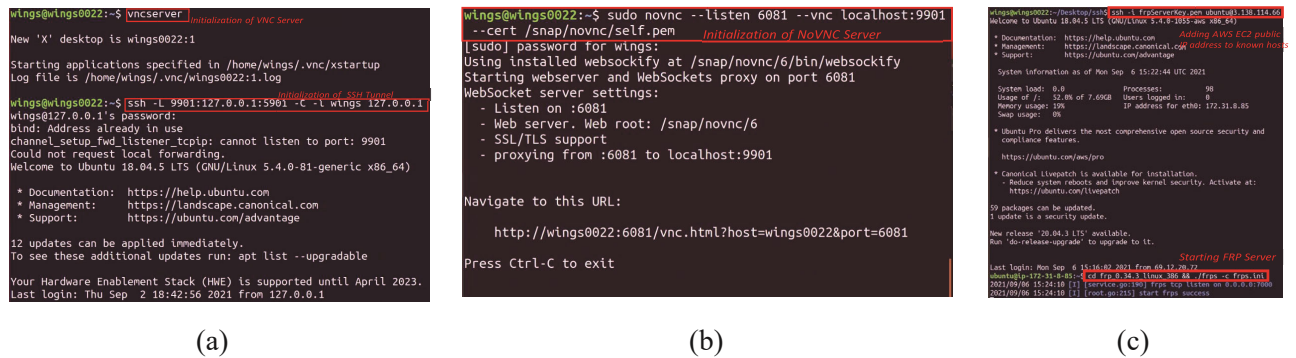


Figure 22: (a) Initializing VNC server and SSH tunnel; (b) initializing NoVNC server; and (c) NoVNC webpage.

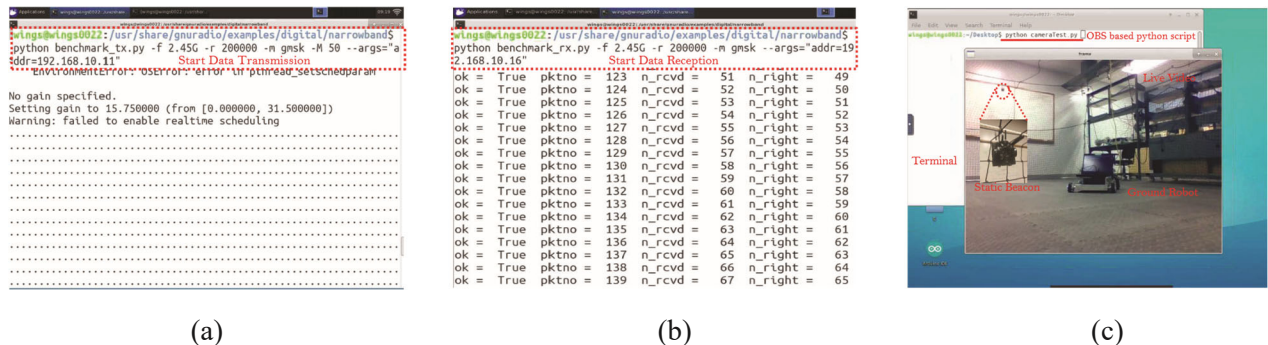


Figure 23: Demonstration of (a) point-to-point data transmission; (b) point-to-point data reception; and (c) online video streaming for remote experiment monitoring.

handle_connect function, which will add the user name and the connection ID of the host to the data table and indicate it as an active user. Similarly, after the experiments, the host can send the disconnect message with action:\$disconnect, and then the handle_disconnect function will be invoked to remove the corresponding entry from the data table. Other messages include action:\$send message and action:\$p2p message with their corresponding functions handle

message and handle_p2pmessage. The former handles all the broadcast messages while the latter deals with point-to-point messages.

Fast Reverse Proxy. It is important for testbed owners to provide secure connections to their local networks that can be accessed by experimenters from outside of their local networks. This can be achieved by implementing a reverse proxy such as Fast Reverse Proxy (FRP) [89]. FRP acts like a tunnel that can redirect the Internet requests from one IP address to another using TCP or UDP connections. This allows us to hide the testbed address and at the same time preserve the integrity of the testbed information.

VNC-Based Remote Command Line. The remote execution of commands from a unified web portal can be enabled by Virtual Network Computing (VNC), a technique that can achieve desktop sharing with low overhead. VNC is platform-independent and follows a client-server-based architecture allowing multiple clients to connect to one VNC server simultaneously without any compromise in performance. VNC is based on an extensible protocol called Remote Frame Buffer protocol (RFB) [90], which allows connections between different versions of VNC clients and servers. A FRP server can be used to tunnel the connection between the public cloud server and the edge server located at the testbed. Before the VNC server connects to the FRP server, an SSH tunnel is established to enhance the security of the connection.

3.3.4 Prototyping and Demonstrations

In this section, we prototype CloudRAFT by integrating UB NeXT, a software-defined testbed for wireless network modeling, optimization and deployment. Next we first give a brief description of the NeXT testbed and then showcase the remote access to the testbed through CloudRAFT.

A snapshot of the NeXT testbed is shown in Fig. 21, where there are two major components: edge server and front-end SDR. The edge server consists of five Dell EMC R340 PowerEdge workstations, each with Intel Xeon E-2246G 3.6 GHz CPU and Ubuntu v18.04. The workstations serve as the controlling hosts of the front-end SDRs for baseband signal processing. The front-end SDR is divided into a *static front-end SDR* and a *mobile frontend SDR*. The former supports experiments with stationary nodes and the latter supports experiments with mobile nodes. For mobile experiments, the SuperDroid programmable robot [91] is used to carry the software radios. In order to control the movement of the ground vehicular robot, we use a Marvelmind super beacon [92] to provide a precise indoor positioning and navigation system. The front-end SDRs consist of 20 N210 and 4 B210 USRPs. The N210 USRPs are powered by three CyberPower PDU41001 Switched Power Distribution Units (PDU). The PDUs allow experimenters to remotely turn on and off the software radios remotely and reboot them in case of technical issues. The edge servers and the front-end SDRs are connected using two switches with Gigabit Ethernet cables.

Establishing Secure Connection. For remote experiments, it is important to establish a secure connection between the experimenter's personal computer and the edge server that controls the front-end SDRs. This can be accomplished following three steps, as shown in Figs. 22(a)-(c).

(i) Initialization of VNC Server and SSH Tunnel. First, the testbed owner starts the secure VNC server using the `vncserver` command. After successful server initialization, an ssh tunnel is created on the edge server that will securely forward the traffic from the VNC server to the edge server. This is done using command `ssh -L 9901:127.0.0.1:5901 -C -l wings 127.0.0.1`, where the `-L` switch specifies the port bindings. In this case, we are binding port 5901 of the remote connection to port 9901 on the local edge server. The `-C` switch enables compression and the `-l` switch specifies the remote login name. A snapshot of the terminal is shown in Fig. 22(a).

(ii) Initialization of NoVNC Server. Second, a NoVNC server is started, which acts as a web-based VNC server. This is achieved using command `sudo novnc --listen 6081 --vnc localhost:9901 --cert/snap/novnc/self.pem`, where `cert` is the certificate for the proxy to encrypt its traffic using WebSocket. Figure 22(b) shows a snapshot of the terminal of this step.

(iii) Initialization of FRP Server and Client. As discussed earlier, the FRP server plays an important role in exposing the local private IP address of the edge server to a public IP address that can be accessed by the testbed users remotely. This is achieved by instantiating an Amazon Elastic Compute Cloud (EC2) instance to create a public IP address and a RSA encrypted private key. Then, the public IP address provided by FRP-EC2 instance is added to the list of known hosts using the following command `ssh -i frpServerKey.pem ubuntu@3.143.218.142`, where `frpServerKey.pem` is the private key. Then, the FRP proxy server can be started using the command `./frps -c frps.ini`. The terminal snapshot is shown in Fig. 22(c). Finally, the FRP client can be started using command `./frpc -c frpc.ini`.

After successful initialization of the FRP server and the client, the remote edge server can be accessed using the following URL: `public ip _address /vnc.html`. The access link along with login credentials will be transmitted to testbed users after an initial request form to ensure the safety and integrity of the testbed operations.

Remote Experiment: Point-to-Point Communications. In this demonstration, we will showcase remote experimentation through CloudRAFT. We consider narrow-band point-to-point communications between two USRP N210 software-defined radios, with IP addresses 192.168.10.11 (tx) and 192.168.10.16 (rx), respectively. Two terminals are opened remotely for tx and rx. The first terminal executes the transmission command `python benchmark_tx.py -f 2.45G -r 200000 -m gmsk -M 50 --args="addr=192.168.10.11"`. Similarly, the second terminal executes the receiving command `python benchmark_rx.py -f python2.45G -r 200000 -m gmsk --args="addr=192.168.10.16"`, where `-f`, `-r`, `-m`, `-M`, `--args` specify the center frequency, transmission rate, modulation scheme, total data size and the address of the USRP devices, respectively. The corresponding snapshot with the packet transmission and reception are shown in Figs. 23 (a) and (b). *Notice that both the transmission and receiving programs are started remotely through the secure link established in the above demonstration.*

Live Monitoring of Remote Experiment. In this demonstration we aim to monitor remotely the movement of the mobile nodes during the experiment. To this end, we stream the live video of the experiment based on a combination of Open Broadcaster Software (OBS) [93] and Real-Time Messaging Protocol (RTMP) [94]. The built-in webcam of Dell Inspiron Laptop is used to capture the live video of the experiment, as shown in Fig. 23(c). To enable indoor navigation of the ground robot vehicle without GPS reception, six static beacons deployed on the net enclosure (the enlarged version of the static beacon is shown in Fig. 23(c)) and one mobile beacon is carried by the robot vehicle.

3.4 FlyBeam: Echo State Learning for Joint Flight and Beamforming Control in Wireless UAV Networks

3.4.1 Background

Unmanned aerial vehicles (UAVs) have been envisioned as an enabling technology for a wide set of new applications, because of their features of fast deployment, high mobility and small size [82,95,96]. Examples of these applications include small cells with flying base stations, UAV-aided guidance, swarm networking for battlefield sensing and data collection, emergency wireless networking in the aftermath of disasters, among others. While UAVs can certainly enable a wide set of new applications, their wide deployment will impose a significant burden on the capacity of the underlying wireless networks. In this work, we focus on designing high-data-rate wireless UAV networks by exploring spatial diversity through collaborative beamforming among the UAVs.

Since it is not easy to mount many antennas on individual UAVs because of their small size, in this work we consider a swarm of UAVs collaborating with each other to perform distributed beamforming. One of the primary challenges is in the formation of UAV clusters for collaborative beamforming. In swarm UAV networks, beamforming can be typically accomplished in two phases. In the first phase, the UAVs estimate the channel state information (CSI) of the wireless links from them to the users they serve and then share the obtained CSI among the UAVs; in the second phase, the UAVs collaborate with each other to perform distributed beamforming for data transmission. While higher beamforming gain can be achieved by forming a larger UAV cluster with more antennas, it takes longer for the CSI sharing in the first phase and hence reduces the time available for data transmission in the second phase. Therefore, a tradeoff needs to be achieved between beamforming gain and channel utilization. Moreover, the beamforming gain is closely coupled with the statistical behaviors of the wireless channels, which are affected by the flight altitude of the UAVs, the dynamic movements of UAVs and the resulting intermittent existence of blockages. Roughly speaking, the wireless communication is dominated by Non-line-of-sight (NLOS) transmissions when the UAVs fly lower, while line-of-sight (LOS) channels are dominant when they fly higher. Additionally, the interference level in the network can be effectively lowered with more and larger blockages, and this also affects the beamforming gain. To account for these coupled factors that jointly affect the beamforming gain in swarm UAV networks, in this report we propose *FlyBeam*, a learning-

based framework for joint flight and beamforming control in swarm UAV networks with unknown blockage dynamics.

3.4.2 Related Work

Wireless UAV networking has drawn significant research attention over the past years [97–102]. For example, in [97] Azari et al. study power control for wireless communications among single-antenna UAVs in cellular networks. In [98] the authors maximize the throughput in UAV-enabled orthogonal frequency-division multiple access (OFDMA) systems with delay-constrained data traffic. The authors of [99] jointly optimize the trajectory and communication in multi-UAV wireless networks to achieve better fairness among users. In [100], we propose a new framework for automated control of swarm UAV networks based on the recent results on principled software-defined wireless networking. Readers are referred to [101,102] and references therein for a good survey of the main results in this area. *Different from these works, which focus on non-collaborative single-antenna UAV communications, in this report we explore spatial diversity in swarm UAV networks by allowing the UAVs to perform distributed beamforming.*

Distributed beamforming has also been extensively studied in wireless networks [103–112]. For example, in [103], Mohanti et al. propose an experimental framework that uses software defined radios to assign beamforming weights that ensure a high level of directivity. In [104], the authors study beamforming vector generation and updating based on recursive channel estimation and updating. The beamforming algorithms for UAV swarms is studied in [105] where the swarm is modeled as a morphing volumetric random array with an assumption that each element is enabled with orientation awareness. However, different from our work, [105] did not consider the dynamic movements and the effects of blockages on the beamforming gain. The authors of [106] present a cooperative communication for cache-enabled UAVs to jointly decide the UAV placement and transmit beamforming based on outdated CSI information. Similar to [105], [106] did not consider the effect of blockages either. In [107] Yuan et al. consider a single UAV-user pair and develop a deep learning-based predictive beamforming scheme that can recover from beam misalignment caused by UAV jittering. Position-based beamforming is studied in [108] to enhance the capacity of UAV communications in LTE networks in the presence of direction-of-arrival estimation errors. In [109], the authors discuss the feasibility and enabling techniques for distributed beamforming in swarm UAV networks. In our previous work [110], we design distributed algorithms for joint power, association and flight control in swarm UAV networks with each UAV endowed with a large number of antennas. Please refer to [111,112] and references therein for an extensive survey of the latest results in this area. *None of these works has studied distributed beamforming in swarm UAV networks by explicitly considering all the factors that affect the beamforming gain, including the flight of UAVs, the resulting dynamic blockages, and CSI sharing among UAVs.*

3.4.3 System Model and Problem Formulation

We consider swarm UAV networks with a set M of single-antenna UAVs collaborating with each other to serve a set U of ground users. The UAVs are allowed to form a virtual MIMO UAV cluster to enhance the quality of the aerial-ground wireless links through distributed beamforming. Focusing on the downlink communications in this setting, our objective is to investigate the effects of those factors that affect the beamforming gain, including blockages in the network, flight altitude of the UAVs as well as the altitude-dependent fading channels, among others. The results obtained in this report can also be extended to uplink scenarios. Next we describe the blockage, channel and beamforming models sequentially.

Blockage Model. Denote S and $|S|$ as the set and the number of blockages in the network, respectively. For each blockage $s \in S$, let C_s, L_s, W_s, H_s and θ_s represent the center, length, width, height and orientation of the blockage, respectively. The orientation θ_s is considered to be uniformly distributed in $[0, 2\pi]$, and the other blockage parameters are randomly generated with the average corresponding to the typical size of blockages in real networks, e.g., the buildings. Let $\mathbf{P}_s^{\text{blk}}(C_s, L_s, W_s, H_s, \theta_s)$ represent the set of points contained in blockage $s \in S$. Denote the location vector of user $u \in U$ as $\mathbf{cod}_u = (x_u, y_u, z_u)$, with x_u, y_u, z_u representing the x-, y- and z-axis coordinates, respectively. Similarly, denote $\mathbf{cod}_m = (x_m, y_m, z_m)$ as the location vector of UAV $m \in M$. Let $\mathbf{P}_{mu}^{\text{link}}$ denote the set of points on the line connecting UAV $m \in M$ and user $u \in U$. Further denote $\mathbf{P}_{mus}^{\text{xt}} = \mathbf{P}_{mu}^{\text{link}} \cap \mathbf{P}_s^{\text{blk}}(C_s, L_s, W_s, H_s, \theta_s)$ as the resulting intersection set of points. Then, given the set S of blockages, the total number of blockages on the link between user u and UAV m , denoted as S_{mu} , can be expressed as

$$S_{mu} = \sum_{s \in S} \mathbf{I}(\mathbf{cod}_m, \mathbf{cod}_u, s), \quad \forall m \in M, \forall u \in U, \quad (9)$$

where $\mathbf{I}(\mathbf{cod}_m, \mathbf{cod}_u, s)$ is the indicator function taking the value of 1 if blockage s is blocking the link and 0 otherwise.

Channel Model. Denote $H_{mu}(f)$ as the path loss for the link between UAV $m \in M$ and ground user $u \in U$ operating in frequency band $f \in F$. Then we model $H_{mu}(f)$ as in [113] as follows:

$$H_{mu}(f) = \eta_0^{S_{mu}} \left(\frac{4\pi f}{C} \right)^2 (d_{mu})^{\alpha_{mu}(f)} \quad (10)$$

where C is the speed of light, $\alpha_{mu}(f)$ is the path-loss exponent for the link between UAV $m \in M$ and ground user $u \in U$ in frequency band f , S_{mu} defined in (9) represents the number of blockages in the link, $\eta_0 \in [0, 1]$ is the per-blockage absorption coefficient [114], and finally $d_{mu} = d_{mu}(\mathbf{cod}_m, \mathbf{cod}_u)$ denotes the distance between UAV $m \in M$ and user $u \in U$.

The transmission time is divided into a set of consecutive time slots. We consider block fading channels in each time slot, i.e., the channel coefficient is considered to be fixed in each

time slot and change to another random value following a certain distribution in the next. Rician fading model is adopted to characterize the fading behavior of the wireless channels, with the Rician factor K depending on whether the link is blocked or not. For NLOS links, i.e. $S_{mu} \neq 0$ in (10), factor K is set to 0; for LOS transmissions, i.e., $S_{m,u} = 0$, factor K is given as $K = 13 - 0.03 \times d_{mu}$ [115]. Denote the resulting channel fading coefficient as h_{mu} for the wireless link between UAV $m \in M$ and ground user $u \in U$.

We further consider pilot-based channel estimation, which can be accomplished in either sequential or simultaneous manner. In the former case, the ground users broadcast their pilot signals in each time slot sequentially, and hence the problem of pilot contamination can be avoided but at the cost of more estimation time. In contrast, with simultaneous channel estimation all the users broadcast their pilot signals at the same time, and hence it takes less time at the cost of lower estimation accuracy. In this report, we take sequential channel estimation as an example, while the results can be easily extended to the simultaneous case. Denote $\mathbf{y}_m = [y_{m1}, \dots, y_{mN_{\text{plt}}}]$ as the pilot signal received by UAV $m \in M$, with y_{mv} , $v = 1, \dots, N_{\text{plt}}$ being the v th symbol of the received pilot signal. Then $\mathbf{y}_m$ can be given as $\mathbf{y}_m = \sqrt{H_{mu}} h_{mu} \mathbf{p}_u + \mathbf{n}_m^0$ where $\mathbf{p}_u = [p_{u1}, \dots, p_{uN_{\text{plt}}}]$ is the pilot sequence of user $u \in U$, and is the vector of Additive White Gaussian Noise (AWGN) at UAV $m \in M$. Let h_{mu} represent the estimated channel gain for the link between UAV $m \in M$ and user. Then, if a least-square estimator [116] is considered, we have

$$\tilde{h}_{mu} = \mathbf{y}_m \times \mathbf{p}_u^\dagger (\mathbf{p}_u \times \mathbf{p}_u^\dagger)^{-1}, \quad (11)$$

where $(\cdot)^\dagger$ denotes the conjugate transpose and $(\cdot)^{-1}$ represents the matrix inverse operation.

Let $t_{\text{est},u}$ denote the time overhead for the channel estimation for user $u \in U$ and t_{est} as the total time overhead for all the users in U . Then we have $t_{\text{est}} = \sum_{u \in U} t_{\text{est},u}$. where $t_{\text{est},u} = N_{\text{plt}}/r$ with N_{plt} being the number of bits in each pilot sequence and r the data rate for pilot transmission.

Beamforming Model. Each UAV $m \in M$ sends its CSI obtained above to a preselected leading UAV of the swarm, denoted as m' with $m' \in \frac{M}{m}$ h, which will then calculate the beamforming weights for the whole swarm. Denote t_{csi} as the resulting time overhead, then we have, $t_{\text{csi}} = \sum_{m \in \frac{M}{m'}} \frac{N_{\text{csi}}}{C'_{mm}}$, where C'_{mm} represents the capacity of the link between UAVs m and m' and N_{csi} is the amount of CSI data in bits to be shared by UAV $m \in M$, i.e., for CSI sharing. Based on the collected CSI, the beamforming weights can be calculated $(\tilde{h}_m = (\tilde{h}_{mu})_{u \in U})$ with h_{mu} obtained in (11). Here, we consider again sequential transmission at the leading UAV. Denote the resulting beamforming weight vector as $\mathbf{w} = (\mathbf{w}_m)_{m \in M}$, where $\mathbf{w}_m = (w_{mu})_{u \in U}$ with w_{mu} being the beamforming weight of UAV m for ground user u . For example, based on *Zero Forcing (ZF)* beamforming [117], $\mathbf{w}_m$ can be given as $\mathbf{w}_m = \mathbf{h}_m^\dagger (\mathbf{h}_m \mathbf{h}_m^\dagger)^{-1}$, where $(\cdot)^\dagger$ denotes the conjugate transpose and $(\cdot)^{-1}$ is the matrix inverse operation. The obtained beamforming weight is then sent back to the corresponding UAVs for the actual

use in the following data transmission. Denote the resulting time overhead as t_{beam} . Then, the overall time overhead denoted as t_{ovhd} can be written as

$$t_{\text{ovhd}} = t_{\text{est}} + t_{\text{csi}} + t_{\text{beam}}, \quad (12)$$

where t_{est} , t_{csi} and t_{beam} are the above-defined time overhead for channel estimation, CSI sharing and beamforming weight feedback, respectively.

FlyBeam Control Problem. For a given frequency f , bandwidth B and coordinates $\mathbf{cod}_u, u \in \mathcal{U}$, as well as the set of blockages $\mathcal{S}$, the control objective of *FlyBeam* is to maximize the aggregate network capacity by jointly controlling the flight and beamforming of the UAVs, as formulated as follows:

$$\begin{aligned} \max \gamma B \sum_{u \in \mathcal{U}} \log_2 \left(1 + \frac{\sum_{m \in \mathcal{M}} P_{mu} \hat{H}_{mu} |\hat{h}_{mu} \hat{w}_{mu}|^2}{\sum_{u' \in \mathcal{U}/u} \sum_{m \in \mathcal{M}} P_{mu'} \hat{H}_{mu'} |\hat{h}_{mu'} \hat{w}_{mu'}|^2 + N_u^0} \right) \\ \text{s.t. : } \sum_{u \in \mathcal{U}} P_{mu} \leq P_{\max}, \forall m \in \mathcal{M}, \end{aligned} \quad (13)$$

where $\hat{H}_{mu} = H_{mu}(\mathbf{cod}_m)$, $\hat{h}_{mu} = h_{mu}(\mathbf{cod}_m)$ and $\hat{w}_{mu} = w_{mu}(\mathbf{cod}_m, \pi)$ denote the path loss, channel fading and beamforming weights for user u and UAV m , respectively, with π denoting the beamforming strategy; P_{mu} and $P_{mu'}$ are the transmission power of UAV $m \in \mathcal{M}$ allocated to users u and u' , respectively; $P_{\max}$ is the maximum transmission power_{slt} of each UAV; N_u^0 is the power of noise at ground user $u \in \mathcal{U}$; and finally $\gamma = \frac{t_{\text{slot}}}{t_{\text{ovhd}}}$ represents the channel utilization coefficient with t_{slot} being the duration of each time slot and t_{ovhd} defined in (12).

3.4.4 FlyBeam Algorithm Design

The primary challenge in solving problem (13) is that the wireless channel hence the beamforming weight w_{mu} and the channel utilization coefficient γ are closely coupled with the UAV location variables $\mathbf{cod}_m$ and hence the resulting dynamic blockages, for which the complete information is unknown to the UAVs. To address this challenge, in this report we solve the problem by designing distributed control algorithms based on a combination of ESN learning [118] and reinforcement learning techniques. Particularly, the ESN is used to approximately model the mapping from the input signals to the output signals of a system, by training its input weights $\mathbf{W}_{\text{in}}$, the reservoir weights $\mathbf{W}$ and output weights $\mathbf{W}_{\text{out}}$ using a sigmoidal transfer function (e.g., hyperbolic tangent). Compared to traditional Neural Networks, which are computationally expensive, it is incredibly simple to train ESNs, while they are still able to model the complex time-varying behaviors of dynamical systems. In this report, we use ESN to approximate the utility function in (13), by jointly considering the UAV flight, the channel estimation and beamforming strategies, as well as the effects of blockages on beamforming. Based on the ESN approximated utility function, online reinforcement learning is then adopted to guide the exploitation and exploration in favor of higher aggregate

capacity. The ESN module consists of four components: *Agent*, *Input*, *Action* and *Reward Function*. In *FlyBeam*, the ESN is implemented in individual UAVs, i.e., each UAV is an *Agent*.

ESN Input Design. Let T denote the total network running time. In each time slot $t \in T$, each UAV $m \in M$ feeds an *Input* (denoted as ρ_m^t) and a candidate *Action* (denoted as ξ_m^t) to its ESN module, which will then output the expected *Reward Function* value of the UAV in the next time slot. For UAV m the input to *FlyBeam*'s ESN module in time slot t , defined as $\rho_m^t \triangleq \{\mathbf{cod}_{-m}^t\}$, comprises of the locations of all the other UAVs $\mathbf{cod}_{-m}^t = (\mathbf{cod}_{m'}^t)$ with $\mathbf{cod}_{m'}^t$ being the coordinate vector of UAV m' in time slot t . The dimension of ρ_m^t increases quadratically with the scale of the network which can slow down the training of *FlyBeam*'s ESN module and hence degrade the utility approximation accuracy in large-scale networks. To address this challenge, the network area is divided into a number $N_x \times N_y \times N_z$ of three-dimensional rectangles, each with $\frac{L_x}{N_x}$, $\frac{L_y}{N_y}$ and $\frac{L_z}{N_z}$ for width, length and height, respectively. Denote N as the set of the resulting rectangles. Each rectangle $n \in N$ is represented using a vector $\mathbf{r}_n = (\widetilde{\mathbf{cod}}_n, \Gamma_n)$, where $\widetilde{\mathbf{cod}}_n$ is the coordinate vector of the center point of rectangle $n \in N$, and $\Gamma_n = 0, 1, \dots, N_x \times N_y \times N_z - 1$ is the index of the rectangle. Based on this policy, the input of *FlyBeam*'s ESN module can be rewritten as $\rho_m^t = (\Gamma_{m'}^t)_{m' \in M/m}$, with $\Gamma_{m'}^t$ is the the index of UAV m' 's rectangle in time slot t .

ESN Action and Reward. Given input ρ_m^t for *FlyBeam*'s ESN module for UAV m in time slot t , UAV m makes its action decisions and observes an output of the action. To this end, UAV m chooses to move to a new rectangle in N except those occupied by other UAVs. The set of actions for UAV m , denoted as ξ_m^t for time slot t , can be written as

$$\xi_m^t = \{\Gamma_v | v \in \mathcal{N} / \{n^t(m')\}, m' \in \mathcal{M}/m\}$$

where $n^t(m')$ represents the rectangle index of UAV m' in time slot t . The corresponding reward is defined as the aggregate network capacity achievable through distributed beamforming as defined by (13). Denote the resulting output capacity as C^t for time slot t .

ESN Training and RL Control. We train the designed ESN module based on online collected capacity data. In the training phase, the objective of the *FlyBeam* ESN module is to learn a model with output C^t that minimizes the root-mean-square error (RMSE) between C^t (i.e., predicted sum capacity) and C_{tgt}^t (i.e., target sum capacity) defined as

$$E(C^t, C_{\text{tgt}}^t) = \frac{1}{N_{\text{out}}} \sum_{i=1}^{N_{\text{out}}} \sqrt{\frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} (C^t - C_{\text{tgt}}^t)^2}, \quad (14)$$

where N_{out} denotes the output units of the ESN, $|\cdot|$ represents the cardinality of a set and T denotes the number of time slots in the training phase. To this end, we measure the sum capacity based on (13) and the measured sum capacity is used as the C_{tgt}^t in (14).

Based on the trained ESN-module, each UAV $m \in M$ can determine its own optimal next-step location m as Γ_m^{t*} . However, this may lead to a local optimum for our swarm control problem defined in (13), which is not desirable. In this report we use reinforcement learning

(RL) to guide the *exploration* and *exploitation* in the flight control of the UAVs. Reinforcement learning [119] has been widely used to solve complex problems that cannot be solved by conventional techniques. In this report, we consider RL algorithm with ϵ -greedy exploration strategy [119]. As the network runs, the online collected data will be used to update the training dataset and then retrain the ESN module. The details of the algorithm for RL control and ESN updating are omitted due to lack of space.

3.4.5 Performance Evaluation

We consider a swarm UAV network with area of $200 \times 200 \times 200 \text{ m}^3$. The frequency and bandwidth are set to 2.1 GHz and 40 MHz, respectively. The transmission power of each UAV is set to 1 W. The LOS and NLOS path-loss exponents are considered to be 2 and 4, respectively. The duration of each time slot is set to 100 ms. Two beamforming schemes are considered: *Zero Forcing (ZF)* and *Maximum Ratio Transmission (MRT)* [117]. The corresponding *FlyBeam* schemes are referred to as *FlyBeam-ZF* and *FlyBeam-MRT*, respectively. We set T and T_{updt} as 3000 and 1000 for initial training and training update phases, respectively. The simulations have been conducted over *SimBAG*, an event-driven Simulator for Broadband Aerial-Ground wireless networks that has been developed in our recent research [82]. Based on SimBAG, a benchmark scheme has been implemented for performance comparison. In the benchmark scheme, no collaborative beamforming is considered, and the UAVs access the channel in a frequency division multiple access (FDMA) manner and further serve their users based on time division multiple access (TDMA). Each UAV maximizes the aggregate capacity of the users it serves based on Sequential Least Squares Programming (SLSQP) [120], to dynamically allocate their transmission power among different sub-channels to avoid mutual interference.

Performance Analysis. In the first experiment, we evaluate the capacity performance of *FlyBeam*. The number of UAVs is varied from 2 to 30 in steps of 2 and the number of ground users is set to 10. The results are reported in Fig. 24. It can be seen from Fig. 24(a) that significant capacity gain can be achieved by *FlyBeam*. For example, average capacities of 915.20 Mbps and 960.12 Mbps can be achieved with *FlyBeam-ZF* and *FlyBeam-MRT*, respectively, which are 3.56 and 3.73 times higher than that achievable by the benchmark scheme (257 Mbps). The effectiveness of distributed beamforming in swarm UAV networks is further verified in Fig. 24(b) in terms of average capacity gain with 10 UAVs and the number of users varying from 2 to 22 in step of 4. We can see that both *FlyBeam-ZF* and *FlyBeam-MRT* are able to achieve a capacity gain between 312% and 457% with an average of 380%.

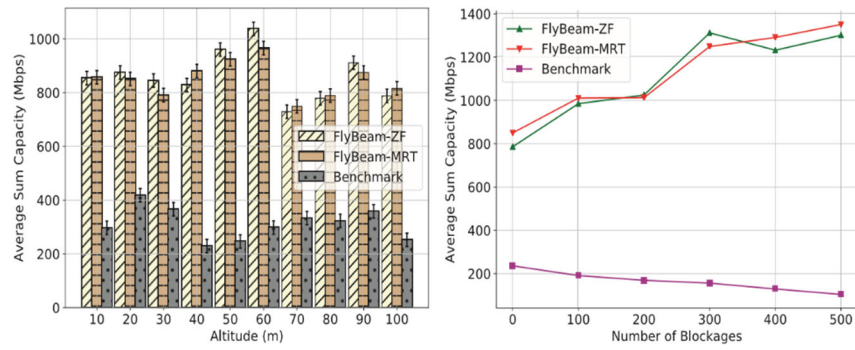


Figure 24: (a) Average sum capacity with different number of UAVs; (b) Capacity gain with different number of users.

Next, we investigate the effects of the UAVs' flight altitude on the beamforming gain, considering 10 ground users and 10 UAVs. The flight altitude of the UAVs is varied from 10 m to 100 m in steps of 10 m. The results are plotted in Fig. 25(a). Similar to that in Fig. 24, significant capacity gains can be achieved by *FlyBeam* in all the tested cases. It can also be found that the sum capacity achievable by *FlyBeam* is only slightly affected by the UAV flight altitude. For example, an average sum capacity of 854 Mbps can be achieved by *FlyBeam-ZF* with a flight altitude of 10 m, which is 816 Mbps for an altitude of 100 m. Similar results can also be achieved by the *FlyBeam-*

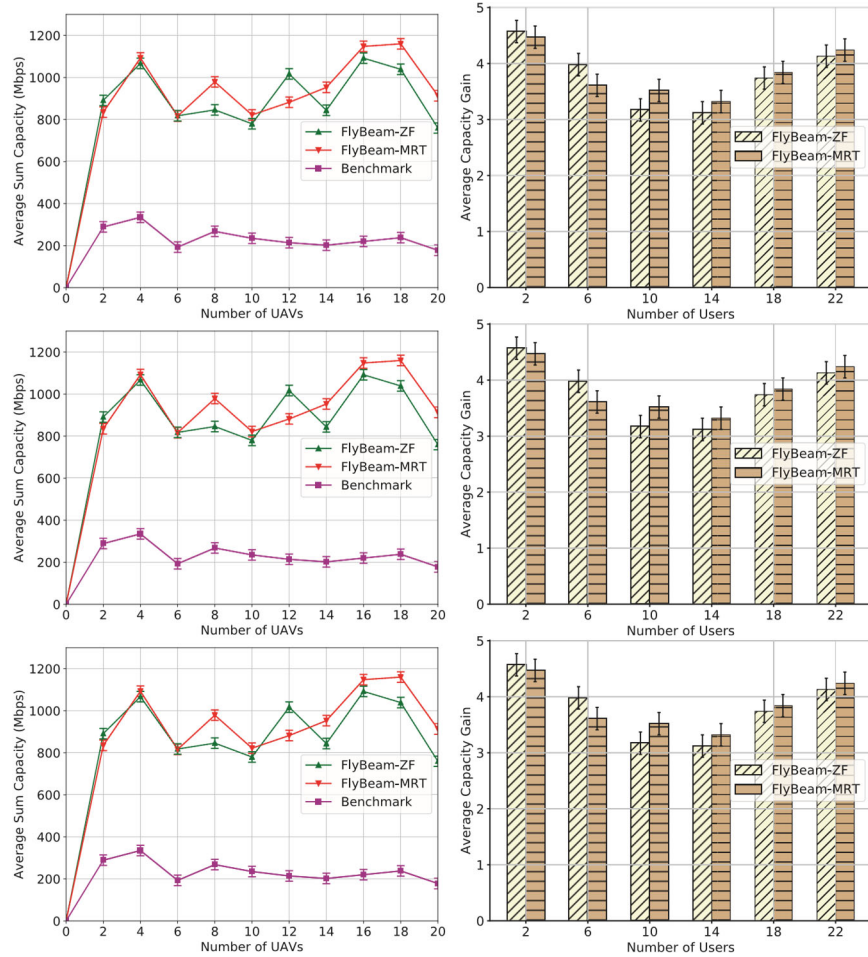


Figure 25: Average sum capacity with (a) varying flight altitude (b) varying number of blockages.

MRT. This is not without reasons. As discussed in Section 7.1, while flying lower can reduce the propagation distance to the ground users, the transmissions may experience higher attenuation because of a higher probability of being blocked. The effects of blockages on the beamforming gain is further studied in Fig. 25(b). The number of blockages is varied from 0 to 500 in steps of 100. The minimum and maximum dimensions of the blockages are set to 2 m and 8 m for length and width, respectively. The height of the blockages is fixed to 15 m, and the UAV flight altitude is set to 30 m.

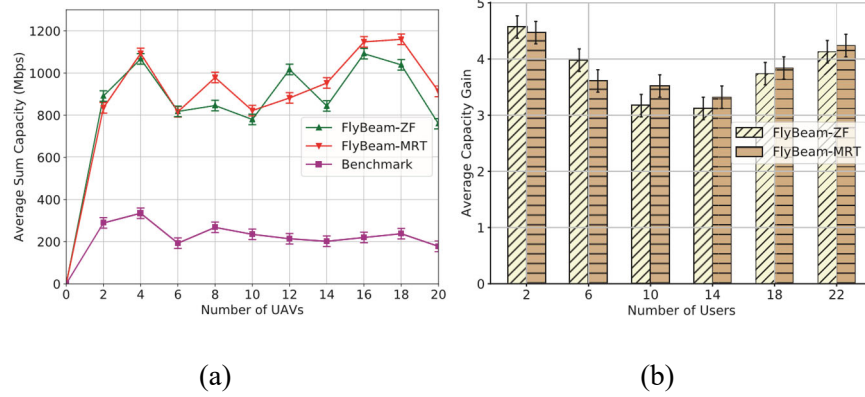


Figure 26: (a) Computational complexity and (b) Communication overhead of FlyBeam.

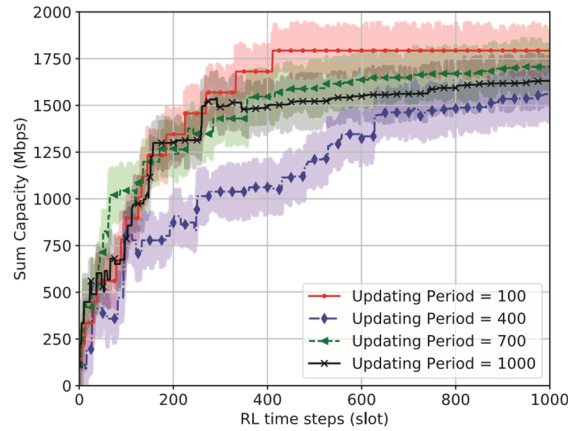


Figure 27: Convergence of FlyBeam with different updating periods.

It was found that *FlyBeam-ZF* and *FlyBeam-MRT* can yield an average capacity of 1106.9 Mbps and 1127.5 Mbps, respectively, which are 6.74 and 6.87 times larger than the benchmark scheme. The sum capacity achievable with the benchmark scheme decreases monotonically with the number of blockages because of larger signal attenuation with more blockages. A surprising observation is that a *higher rather than lower* sum capacity can be achieved by *FlyBeam* with more blockages in the network. For example, an average sum capacity of 786 Mbps can be achieved by the *FlyBeam-ZF* with no blockages, which are 1025 Mbps and 1301 Mbps with 200 and 500 blockages, respectively. This is primarily because denser blockages introduce more diversity in the wireless channels, and hence higher spatial diversity gain can be achieved through beamforming.

Complexity and Convergence. We study the computational complexity and communication overhead of *FlyBeam* control in Fig. 26. The experiments are conducted on a workstation with

Intel(R) Core (TM) i7 – 10510U CPU @ 1.80 GHz 2.30 GHz, memory of 16.0 GB, and 64-bit Windows Operating System. We fix the number of users as 10 and vary the number of UAVs from 3 to 10 at a step of 1. In Fig. 26(a) we report the average computational time taken by the UAVs' *FlyBeam* ESN module, with the confidence interval indicated by the shaded area. It can be seen that the computational time ranges from 0.14 ms to 0.21 ms and is barely affected by the network scale. Figure 26(b) plots the corresponding communication overhead, i.e., the time taken for CSI and beamforming weights sharing. Results indicate that the communication overhead of individual UAVs varies only slightly with the number of UAVs, from 0.43 ms to 0.46 ms with an average of 0.443 ms. This verifies the good scalability of *FlyBeam*.

We further study the effects of periodic updating of the ESN training on the network's sum capacity, taking *FlyBeam*-MRT as an example. The results are plotted in Fig. 27, where each curve is obtained by averaging over 20 simulations. The updating period is varied from 100 to 1000 time slots at steps of 300 time slots. It can be seen that the convergence speed is significantly affected by the updating period. In most of the test cases, *FlyBeam* converges faster with shorter updating periods. For example, *FlyBeam* converges in around 400 time slots with an updating period of 100 time slots, which is twice as fast as with updating periods of 700 and 1000 time slots. However, interestingly, we found that there is no monotonic mapping between the convergence speed and the updating period. This can be seen when the updating period is set to 400 time slots. In future work, we will investigate the optimal updating frequency for online reinforcement learning by considering the involved interaction between ESN-based utility function approximation and RL-based state exploration.

3.5 SwarmShare: Mobility Resilient Spectrum Sharing

3.5.1 Motivation

Unmanned aerial vehicles (UAVs) have been envisioned as a key technology for next-generation (i.e., B5G or 6G) wireless networks [121,122]. Because of their features of fast deployment, high mobility and small size, UAVs have a great potential to enable a wide set of new applications, including UAV-aided guidance, small cells with flying base stations, emergency wireless networking in the aftermath of disasters, among others. The foreseen wide adoption of UAV systems can pose a significant burden on the capacity of the underlying wireless networks. In this report, we aim to explore new approaches that can enable UAV operations in the 6 GHz band to harvest the additional 1.2 GHz spectrum bandwidth [123].

The primary challenge towards this goal is in the spectrum sharing approaches adopted by the incumbent systems in this frequency band. The 6 GHz band consists of four sub-bands, i.e., U-NII-5 (5.925-6.425 GHz), U-NII-6 (6.425-6.525 GHz), U-NII-7 (6.525-6.875 GHz), and U-NII-8 (6.875-7.125 GHz). These bands have been previously occupied by a set of nongovernment services, including fixed point-to-point services, fixed-satellite service (Earth-to-Space), broadcast auxiliary service and cable television relay service [123].

These incumbent systems coexist with each other by sharing the spectrum on a directional basis, i.e., they use highly directional antennas to concentrate the signal energy in a particular direction such that mutual interference can be effectively mitigated as long as their antennas are not pointed toward each other. As a result, traditional carrier-sensing-based spectrum sharing as in Wi-Fi networks is non-applicable to extend those wireless systems with omnidirectional antennas to this frequency band, because of the low detectability of the incumbent systems. For this reason, two operation modes have been proposed by the FCC, i.e., standard-power and low-power modes. The former allows both indoor and outdoor operations on the U-NII-5 and U-NII-7 bands with maximum transmission power of 30 dBm. The latter focuses on indoor operations in the U-NII-6 and U-NII-8 bands with maximum transmission power of 24 dBm.

However, none of the above two modes support UAV operations in the 6 GHz bands [123,124]. A major concern is with the high mobility of the UAV systems, which makes it difficult to model and control their aggregate interference to the incumbent systems. The situation gets even worse when considering the altitude-dependent interference range of UAVs and the higher probability of line-of-sight signal propagation at higher altitudes. Additionally, it is also challenging for the distributed UAVs to control their aggregate interference collaboratively by jointly considering their spectrum access strategies and association to the ground users.

To address these challenges, a key step is to understand the statistical behaviors and effects of the aggregate interference experienced by the incumbent systems, since no real-time cross-system channel state information (CSI) is available. To this end, in this report we focus on a new spectrum sharing scenario in the 6 GHz band called *SwarmShare*, where a set of UAVs collaboratively provide data streaming services to ground users, by sharing the spectrum with the incumbent systems on the 6 GHz band under the cross-system interference constraints. Within this framework, we model and analyze the aggregate interference from the UAVs to the incumbent, and propose a mobility-resilient stochastic spectrum sharing approach.

3.5.2 Related Work

UAV systems have attracted significant research attention in both academia and industry [121, 125–128]. For example, in [121] the authors optimize the achievable rate of UAV-aided cognitive IoT networks. Wang et al. propose in [125] a dynamic hyper-graph coloring approach for spectrum sharing in UAV-assisted networks. In [126], the authors optimize mobile terminals' throughput by jointly controlling UAV trajectory, bandwidth allocation and user partitioning between the UAV and ground base stations. In [127], UAV is used as a relay to assist D2D communications. [128] studies machine learning based spectrum sharing for UAV-assisted emergency communications. Readers are referred to [129] and references therein for a survey of the main results in this area.

Spectrum sharing in cognitive radio networks has also been a hot research topic for a long time with a sizable and increasing body of literature. In [130], the authors aim to maintain network connectivity in cognitive radio networks by controlling the transmission power of sensors. In [131], the authors maximize the revenue of the newly joined systems in cognitive radio networks by controlling the channel access of new users. The authors of [132] propose a cognitive backscatter network to maximize the data rate of the newly joined networks.

Spectrum sharing between directional- and omnidirectional-antenna wireless systems has also been studied in existing literature. For example, [133] optimizes the performance of LTE-Unlicensed networks while guaranteeing the performance of the co-located radar system. The authors of [134] propose RadChat, a distributed networking protocol for mitigation of interference among frequency-modulated continuous wave radars. A cooperative spectrum sharing model is proposed in [135] to mitigate the mutual interference among radar and communication systems. Please refer to [136] and references therein for a good survey of the main results in this field. Different from the above-discussed works, none of which have considered the spectrum sharing between UAVs and the incumbent wireless systems in the 6 GHz band, in this report we aim to *design a new, mobility-resilient spectrum sharing framework to enable wireless UAV networking in this band*.

3.5.3 System Model and Problem Formulation

We consider a wireless UAV network coexisting with an incumbent communication Tx and Rx pair by sharing the same portion of spectrum B in the 6 GHz band. The UAV network consists of a set K of UAVs collaborating with each other to serve a set M of ground users. The transmission time is divided into a set T of consecutive time slots. In each time slot $t \in T$, denote the coordinate vector of UAV $k \in K$ as $\mathbf{cod}_k^t = [x_k^t, y_k^t, z_k^t]^T$, with $\mathbf{T}$ being the transpose operation and x_k^t , y_k^t and z_k^t representing the x-, y- and z-axis components, respectively. Similarly, denote respectively $\mathbf{cod}_{Tx} = [x_{Tx}, y_{Tx}, z_{Tx}]^T$, $\mathbf{cod}_{Rx} = [x_{Rx}, y_{Rx}, z_{Rx}]^T$ and $\mathbf{cod}_i = [x_i, y_i, z_i]^T$ as the coordinate vectors of incumbent transmitter Tx, incumbent receiver Rx and ground node $i \in M \cup \{Tx, Rx\}$. Denote $A \in K \cup M \cup \{Tx, Rx\}$ as the set of all the nodes in the heterogeneous network. The objective of the UAV network is to maximize its own spectral efficiency under the interference constraints of the incumbent system. Before presenting the formal formulation of the spectrum sharing problem, we first describe the considered channel, antenna and throughput models.

Channel Model. We consider both large-scale path-loss and small-scale fading. For path-loss, we consider line-of-sight (LoS) wireless channels between the incumbent transmitter Tx and its receiver Rx. This is feasible because the incumbent systems are usually carefully deployed such that their antennas are well-aligned without any obstructions in the link. However, we consider non-line-of-sight (NLoS) links between the incumbent nodes and

the ground users of the coexisting networks. For UAV network, we consider as in [137] a probabilistic path-loss model for the links between UAVs and ground nodes. Then, the LoS and NLoS path-loss (in dB) between UAV $k \in K$ and ground node $i \in M \cup \{\text{Tx}, \text{Rx}\}$ can be given as, in time slot $t \in T$

$$H_{k,i}^{\text{LoS},t} = 20 \log \left(\frac{4\pi d_{k,i}^t f}{c} \right) + \eta^{\text{LoS}}, \quad (15)$$

$$H_{k,i}^{\text{NLoS},t} = 20 \log \left(\frac{4\pi d_{k,i}^t f}{c} \right) + \eta^{\text{NLoS}}, \quad (16)$$

where the first item on the right-hand side of (15) and (16) represents the free space pathloss with $d_{k,i}^t = \|\mathbf{cod}_k^t - \mathbf{cod}_i\|_2$ being the distance between UAV k and receiver i in time slot t , f is the carrier frequency of UAV k , c is the speed of light, and η^{LoS} and η^{NLoS} are the additional attenuation factors due to LoS and NLoS transmissions, respectively. Let $\Pr(H_{k,i}^{\text{LoS},t})$ represent the probability of LoS transmissions in time slot t , then $\Pr(H_{k,i}^{\text{NLoS},t})$ can be expressed as [138],

$$\Pr(H_{k,i}^{\text{LoS},t}) = \left(1 + X \exp(-Y(\phi_{ki} - X)) \right)^{-1} \quad (17)$$

Finally, for small-scale fading we consider Rician fading for LoS transmissions and Rayleigh fading for NLoS. Denote K_{ij} as the Rician factor for the wireless channel between nodes $i, j \in A$, then $K_{ij} \in A$ can be given as $K_{ij} = 13 - 0.03d_{ij}$ for LoS transmissions and 0 for NLoS. Denote the resulting small-scale fading coefficient as $h_{ij}^t \triangleq h_{ij}^t(K_{ij})$ for nodes $i, j \in A$.

Antenna Model. We consider directional transmissions for the incumbent wireless systems and omnidirectional transmissions for the coexisting UAV network. Specifically, we consider as in [82] bi-sectorized antenna model to characterize the interference between directional and omnidirectional antennas. Denote θ_{Tx} and θ_{Rx} as the signal beamwidth of the incumbent transmitter and receiver's antennas, respectively. Let $\theta_m \in [-\pi, \pi]$ denote the offset angle of the boresight direction of the Tx's antenna with respect to the reference direction for ground user $m \in M$. Here, the reference direction refers to the direction along which the Tx's antenna would be exactly pointed to user m . Then the antenna gain of incumbent transmitter Tx with respect to ground user $m \in M$ in time slot t , denoted as $w_{m\text{Tx}}^t$, can be written as

$$w_{m\text{Tx}}^t = \begin{cases} w_{m\text{Tx}}^{\max}, & \text{if } \theta_m \leq \theta_{\text{Tx}} \\ w_{m\text{Tx}}^{\min}, & \text{otherwise} \end{cases} \quad (18)$$

Where $w_{m\text{Tx}}^{\max}$ and $w_{m\text{Tx}}^{\min}$ represent the maximum and minimum transmit gains of the incumbent transmitter, respectively. Similarly, the receive gain of the incumbent receiver Rx with respect to UAV $k \in K$, denoted as $w_{m\text{Rx}}^t$, can be given as

$$w_{mRx}^t = \begin{cases} w_{mRx}^{\max}, & \text{if } \theta_m \leq \theta_{Rx} \\ w_{mRx}^{\min}, & \text{otherwise} \end{cases} \quad (19)$$

with $w_{mRx}^{\max}$ and $w_{mRx}^{\min}$ being the maximum and minimum receive gains of the incumbent receiver, respectively. The transmit and receive gains are set to the maximum values for incumbent transmissions, i.e., $w_{mTx}^{\max}$ and $w_{mRx}^{\max}$, respectively.

Throughput Model. Based on the above channel and antenna models, the signal-to-interference-plus-noise ratio (SINR) of the incumbent receiver Rx, denoted as γ_{Rx}^t for time slot t , can be written as

$$\gamma_{Rx}^t = \frac{p_{Tx} w_{mTx}^{\max} w_{mRx}^{\max} (h_{TxRx}^t)^2 / H_{TxRx}^{\text{LoS}}}{\sum_{k \in K} p_k^t w_{kRx}^t (h_{kRx}^t)^2 H_{kRx}^{\text{LoS}} + (\sigma_{Rx})^2} \quad (20)$$

where p_{Tx} and p_k^t represent the transmission power of the incumbent transmitter Tx and UAV $k \in K$ in time slot $t \in T$, respectively; w_k denotes the transmit gain of the UAV and is considered to be constant for omnidirectional antennas; and $(\sigma_{Rx})^2$ is the power of Additive White Gaussian Noise (AWGN) at the incumbent receiver.

The objective of *SwarmShare* is to guarantee satisfactory SINR for the incumbent system (i.e., γ_{Rx}^t above) by controlling the transmission power of the coexisting UAVs. To this end, we consider a single-home association strategy for the ground users of the UAV network, that is in each time slot $t \in T$ each ground user can be served by at most one UAV. Denote α_{km} as the association variable, with $\alpha_{km} = 1$ if ground user $m \in M$ is associated with UAV $k \in K$ and $\alpha_{km} = 0$ otherwise. Then we have

$$\sum_{k \in K} \alpha_{km}^t \leq 1, \forall k \in K, m \in M, t \in T \quad (21)$$

$$\alpha_{km}^t \in \{0, 1\}, \forall k \in K, m \in M, t \in T \quad (22)$$

We further consider FDMA-based spectrum access among the UAVs in K and TDMA for the ground users served by the same UAV. Then, the SINR of ground user $m \in M$ in time slot t , denoted as $\gamma_m^t = \gamma_m^t(H_{mTx}^t)$ can be expressed as

$$\gamma_m^t = \frac{p_{k(m)}^t (h_{k(m)m}^t)^2 / H_{k(m)m}^t}{(p_{Tx}/|K|) w_{mTx}^t \hat{w}_m (h_{mTx}^t)^2 / H_{mTx}^t + (\sigma_m)^2} \quad (23)$$

Notice in (23) that only $1/|K|$ of the incumbent transmitter's power (i.e., $p_{Tx}/|K|$) is considered for each UAV and its associated ground users because of the UAVs' FDMA-based spectrum access. It is worth pointing out that we consider FDMA- and TDMA-based spectrum access for the UAV networks because we want to focus this work on the interference control between the UAV and the incumbent systems. The resulting cross-system spectrum sharing scheme can also be extended to other more advanced spectrum access schemes for UAVs [139,140].

Finally, the capacity achievable by user m in time slot t , denoted as $C_{k(m)m}^t$, can be expressed as

$$C_{k(m)m}^t = \frac{B}{|K||M_k^t|} \left[\Pr(H_{k(m)m}^{\text{LoS},t}) \log_2(1 + \gamma_m^t(H_{k(m)m}^{\text{LoS},t})) + \Pr(H_{k(m)m}^{\text{NLoS},t}) \log_2(1 + \gamma_m^t(H_{k(m)m}^{\text{NLoS},t})) \right] \quad (24)$$

where $\Pr(\cdot)$ is the probability of LoS and NLoS transmissions defined in Section 7.3.1 and $\gamma_m^t(\cdot)$ is the SINR of ground user m defined in (23).

Problem Formulation. The objective of the *SwarmShare* control problem is to maximize the aggregate capacity of the UAV network by jointly controlling the transmission power of the UAVs and their flight trajectory as well as association with the ground users, while meeting the cross-system interference constraints, as formulated as

$$\max_{\mathbf{P}, \mathbf{A}, \mathbf{Q}} \frac{1}{|T|} \sum_{t \in T} \sum_{m \in M} C_{k(m)m}^t \quad (25)$$

$$\text{Subject to: } 0 \leq p_k^t \leq p_{\max}, \forall k \in K, t \in T \quad (26)$$

$$\text{Association Constraints (21)(22)} \quad (27)$$

$$\frac{1}{|T|} \sum_{t \in T} \mathbb{I}(\gamma_{\text{Rx}}^t \leq \gamma_{\text{Rx}}^{\text{th}}) \leq \Pr_{\text{Rx}}^{\max} \quad (28)$$

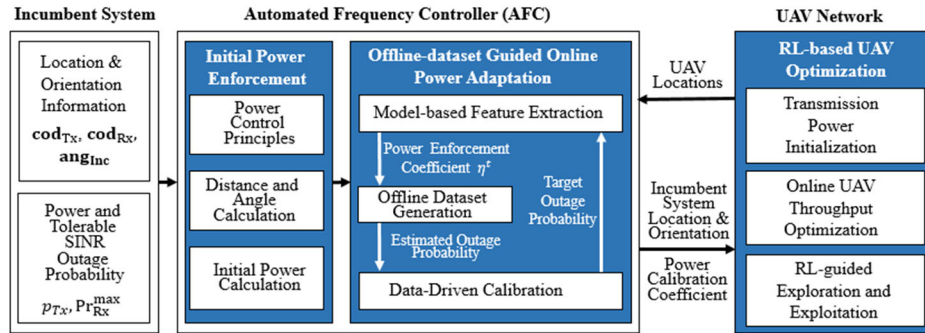


Figure 28: *SwarmShare* Spectrum Sharing Framework.

3.5.4 Spectrum Coexistence Design

The *SwarmShare* problem formulated in (25)-(28) is a mixed integer nonlinear nonconvex programming (MINLP) problem, because of the binary UAV-user association variables α_{km}^t and the underlying complicated mathematical expressions in (25) and (28). Moreover, to solve the problem directly it requires to know the real-time channel state information (CSI)

between the UAV network and the incumbent system, which is however unavailable, as discussed in Section 7.1, because of the low-detectability of the directional incumbent signals.

To address the above challenges, in this work we consider an AFC (Automated Frequency Controller)-assisted spectrum sharing. AFC has been adopted for spectrum sharing in the TV whitespace band as well as the 6 GHz band by determining certain exclusion zones nearby the incumbent systems [123]. Our work differs from this with our objective to enable exclusion-zone-free hence more flexible spectrum sharing, and study the statistical behavior of the aggregate interference from the UAV networks to the incumbent system, while keeping the cross-system signaling at a minimum level. The diagram of the proposed spectrum coexistence framework is illustrated in Fig. 28, where there are three major components, i.e., *Initial Power Enforcement*, *Offline-dataset Guided Online Power Adaptation*, and *Reinforcement Learning-based UAV Optimization*.

Initial Power Enforcement. The objective of this phase is to determine, following a set of *Power Control Principles*, a rough transmission power for each of the UAVs. In this work, we consider three basic principles to accommodate the effects of the UAVs' flight altitude and their locations on the interference to the incumbent system, while more sophisticated principles can be incorporated in the future. These principles are i) UAVs that are closer to the incumbent receiver should transmit at lower power; (ii) with the same distance to the incumbent receiver, UAVs flying higher should transmit at lower power; and (iii) with the same distance and altitude, UAVs with smaller angles relative to the boresight axis of the incumbent receivers' directional antenna should transmit at lower power. Particularly, the rationale of the second principle is that, with the hybrid LoS/NLoS channel model described in Section 7.3.1, it is more likely for a UAV to establish LoS links to the incumbent receiver when flying higher and hence cause more interference. Similarly, for the third principle, based on the directional antenna model described in Section 7.3.2, a UAV will cause higher interference when more aligned with the incumbent receiver's antenna.

In SwarmShare, an initial power enforcement coefficient $\text{Enf}(\mathbf{cod}_k^t, \mathbf{cod}_{\text{Rx}}, \mathbf{ang}_{\text{Inc}})$ will be calculated for each UAV $k \in K$ in time slot $t \in T$ based on the above three principles. This is accomplished using three Sigmoid-family functions $\text{Sig1}(\cdot)$, $\text{Sig2}(\cdot)$ and $\text{Sig3}(\cdot)$ as follows

$$\begin{aligned} \text{Enf}(\mathbf{cod}_k^t, \mathbf{cod}_{\text{Rx}}, \mathbf{ang}_{\text{Inc}}) = & \text{Sig}_1\left(\frac{l_{\text{euc}}(\mathbf{cod}_k^t, \mathbf{cod}_{\text{Rx}})}{l_{\text{euc}}^{\text{th}}}\right) + \text{Sig}_2\left(\frac{h(\mathbf{cod}_k^t) + h(\mathbf{cod}_{\text{Rx}})}{l_{\text{hgh}}^{\text{th}}}\right) \\ & + \text{Sig}_3(l_{\text{rad}}(\mathbf{cod}_k^t, \mathbf{cod}_{\text{Rx}}, \mathbf{ang}_{\text{Inc}})) \end{aligned} \quad (29)$$

With the obtained power enforcement coefficient, each UAV's power can be initialized as, in time slot $t \in T$,

$$p_k^{\text{ini}} = p^{\text{max}} \text{Enf}(\mathbf{cod}_k^t, \mathbf{cod}_{\text{Rx}}, \mathbf{ang}_{\text{Inc}}) \quad (30)$$

Offline-dataset Guided Online Power Adaptation. Our goal is to enable UAV operations in the 6 GHz band while meeting the cross-system interference constraint (28). In *SwarmShare*, this is accomplished by fine-tuning the above-obtained initial transmission powers for the UAVs following a three-step approach, as described as follows.

Model-based Feature Extraction: In this step, we first extract the network features that can be used later in *Data-Driven Calibration*, rather than using directly the raw network topology information such as UAV location vector $(\mathbf{cod}_k^t)_{k \in \mathcal{K}}^{t \in \mathcal{T}}$. This is important to mitigate the curse of the dimensionality problem [141] especially with a large number of UAVs. In *SwarmShare*, we select the power adaptation coefficient, denoted as η^t for time slot t , as the network feature. Then, given the above obtained initial transmission power p_k^{ini} for UAV $k \in K$, a new transmission power p_k^t can be calculated as

$$p_k^t = p_k^{\text{ini}} \eta^t \quad (31)$$

and interference constraint (28) can be rewritten as

$$\frac{1}{|T|} \sum_{t \in T} \mathbb{I}(\gamma_{\text{Rx}}^t(\eta^t) \leq \gamma_{\text{Rx}}^{\text{th}}) \leq \text{Pr}_{\text{Rx}}^{\text{max}}, \quad (32)$$

where $\gamma_{\text{Rx}}^t(\eta^t)$ is the SINR of the incumbent receiver defined in (20) by substituting (31) into (20). Consider the ergodic stochastic process for the aggregate interference and denote $\text{Prob}(\gamma_{\text{Rx}}^t(\eta^t) \leq \gamma_{\text{Rx}}^{\text{th}})$ as the SINR outage probability in time slot $t \in T$, then the left-hand side of (32) can be equivalently represented as

$$\begin{aligned} \text{Prob}(\gamma_{\text{Rx}}^t(\eta^t) \leq \gamma_{\text{Rx}}^{\text{th}}) &= \text{Prob}\left(\frac{P_{\text{Rx}}^{\text{sig}}}{P_{\text{Rx}}^{\text{itf}}} \leq \gamma_{\text{Rx}}^{\text{th}}\right) \\ &= \int_0^{+\infty} \int_{\frac{P_{\text{Rx}}^{\text{th}}}{\gamma_{\text{Rx}}}}^{+\infty} \text{pdf}_{P_{\text{Rx}}^{\text{sig}}}(p^{\text{sig}}) \cdot \text{pdf}_{P_{\text{Rx}}^{\text{itf}}}(p^{\text{itf}}) dp^{\text{itf}} dp^{\text{sig}} \end{aligned} \quad (33)$$

where $P_{\text{Rx}}^{\text{sig}}$ and $P_{\text{Rx}}^{\text{itf}}$ are the numerator and denominator of (20), respectively; Rayleigh distribution has been considered for the small-scale fading and hence non-central chi-square distribution [142] for the receive power of the incumbent signals; and finally as in [143, 144] Gamma distribution is considered for the aggregate interference power. The details of mathematical equations are omitted for the distributions due to space limitations. It is worth pointing out that the aggregate interference of UAVs does not follow any existing statistical distributions. In this work, we consider Gamma distribution in (35) because we want to obtain a rough estimation of the power adaptation coefficient η^t , which will be further calibrated based on an offline dataset. Notice that given the maximum tolerable SINR outage probability $\text{Pr}_{\text{Rx}}^{\text{max}}$ in (32), the maximum η^t can be determined efficiently by

bisection search, since the left-hand side of (32), which is equivalent to (35), is a monotonically increasing function of the UAVs' transmission power hence η^t .

Offline-dataset Generation: Given the above-obtained network feature η^t , and each UAV's transmission power p_k^t can be updated according to (31). Since the power adaptation may be inaccurate because of the inaccuracy of the Gamma distribution-based interference model in (35), we further calibrate the power control for UAVs with the assistance of offline measurements. Specifically, given the transmission power vector $(p_k^t)_{k \in K}$, the corresponding SINR outage probability of the incumbent system can be obtained by offline simulations. By varying the number of UAVs, their locations as well as the maximum tolerable SINR outage probability in the simulations, we are able to obtain an SINR outage probability vector.

Data-Driven Calibration: Finally, a mapping can be established through function approximation, e.g., based on linear regression [145], echo state learning [146] or deep neural networks [147]. In this work we find that it is enough to approximate the mapping based on linear regression

Reinforcement Learning-based UAV Optimization. As illustrated in Fig. 28, the above-obtained η^t will be broadcast to the UAVs, which will then calculate their transmission power based on (31). Meanwhile, the UAVs will update their flight and association strategies to serve their users with higher spectral efficiency. To this end, we consider as in [82] shortest-distance-based association strategy. Then, in each time slot $t \in T$ the association variables $\alpha_{k,m}$, defined in Section 7.3 can be determined as

$$\alpha_{k,m} = \begin{cases} 1, & \text{if } k = \operatorname{argmin}_k \|\mathbf{cod}_k^t - \mathbf{cod}_m\| \\ 0, & \text{otherwise} \end{cases} \quad (34)$$

Further divide the whole network area into a set of three-dimensional rectangles. In each time slot, each UAV is allowed to either move to one of its adjacent rectangles or stay in the current. For each of the candidate rectangles, the UAV will first calculate the achievable capacity given the transmission power calculated in Sections 7.4.1 and 7.4.2 and the set of ground users it serves. Finally, as in [148], reinforcement learning with -greedy search is adopted to guide the exploitation and exploration during the UAV's flight control. The details of the learning algorithm are omitted due to space limitations.

Complexity Analysis. The most time-consuming operation in the above procedure is offline-dataset guided online power adaptation in Section 7.4.2. In Section 7.4.2, we need to determine the network feature η^t given the UAV network's topology and the maximum tolerable SINR outage probability. Since the SINR outage probability is a monotonically increasing function of η^t , the value of η^t can be obtained efficiently based on bisection search. The dataset generation and regression can be conducted offline, and $\Pr_{\text{Rx}}^{\max}$ can be determined online by solving a linear problem.

Regarding communication overhead, as illustrated in Fig. 28, the AFC needs to collect one-time location and orientation information of the incumbent system and broadcast the collected information to the UAVs. If the incumbent system does not move frequently (which is usually the case, e.g., fixed point-to-point applications), the resulting communication overhead can be neglected. The AFC also needs to collect periodically the

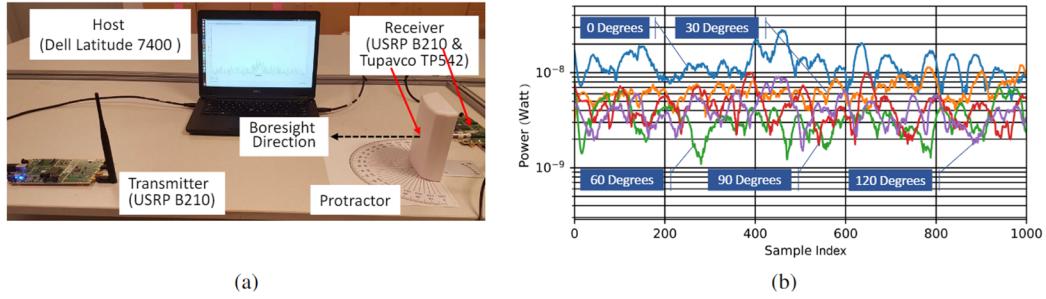


Figure 29: (a) Snapshot of the testbed setup for threshold angle measurement; (b) Examples of the measurement results.

UAVs' locations and broadcast the updated power adaptation coefficient η^t to the UAVs. Since it is enough to represent these information in 16 bytes (three float numbers for location and one for the power adaptation coefficient, and each float number takes 4 bytes), the resulting broadcast overhead is low as well. Moreover, the UAVs do not need to report their locations to the AFC in real-time, without obviously increasing the SINR outage probability of the incumbent system. This will further reduce the communication overhead.

3.5.5 Performance Evaluation

In this section we validate the effectiveness of the *SwarmShare* framework. We consider a network area of $500 \times 500 \times 50 \text{ m}^3$, with 50 ground users randomly located in the network and the number of UAVs varying from 3 to 24. The incumbent transmitter and receiver are deployed with coordinates of (200, 200, 10) and (250, 250, 10), respectively. The center frequency of the shared spectrum is set to 6 GHz with total bandwidth of 10 MHz. The maximum transmission power of the incumbent transmitter and the UAVs are set to 1 W and 0.25 W, respectively. For the bisectorized antenna model, the maximum and minimum gains are set to 1 and 0.5, respectively. The power density of the AWGN is set to -174 dBm/Hz. The probability of LoS and NLoS links are set to 0.7 and 0.3, respectively. The threshold parameters $l_{\text{euc}}^{\text{th}}$ and $l_{\text{hgh}}^{\text{th}}$ in (29) are set to 70 m and 30 m, respectively. Next, before discussing the interference control results, we first determine the threshold angle for the directional antenna model and validate the effectiveness of the data-driven calibration scheme.

Threshold Angle Measurement. We first determine the threshold angle for the directional antenna model by conducting a set of experimental measurements. A snapshot of the testbed is shown in Fig. 29(a), where the transmitter is a USRP B210 software radio with an omnidirectional antenna, the receiver is another USRP B210 with Tupavco TP542 antenna, and the baseband signal processing is conducted based on GNU Radio on a Dell Latitude 7400 laptop. Tupavco TP542 is a directional Wi-Fi antenna operating in the frequency range up to 5.8 GHz (very close to the 6 GHz band) with antenna gain of 13 dB. We measure the received power by varying the relative of the transmitter with respect to the boresight direction of the directional antenna (as illustrated in Fig. 29 (a)) and the transmission distance from 1 to 3 meters. Examples of the measurement results are given in Fig. 29(b) with transmission range of 1 m and relative angles varying from 0 to 120 degrees at steps of 30 degrees. The mapping between the received power and relative angle is established based on logarithmic regression method [149]. Based on the regression results, we set 30 degree as the threshold angle for the bisectorized antenna model, which corresponds to the 3 dB angle of the Tupavco TP542 antenna.

Data-driven Interference Prediction. Given the above obtained threshold angle, we further characterize the statistical behavior of the aggregate interference from the coexisting UAVs to the incumbent receiver. To this end, we conduct a set of simulations over UBSim, a newly developed Universal Broadband Simulator for integrated aerial-ground networking. Details of the simulator are omitted due to space limitations. The results are reported in Figs. 30(a) and (b) with 10 and 20 UAVs, respectively. We fit as in [150,151] the collected interference values using four distributions, including Gaussian, Inverse Gaussian, Gamma and Inverse Gamma, and find that the power of the aggregate interference does not follow any of these distributions. This is actually our motivation to design *SwarmShare* based on a data-driven approach. Fig. 30(c) reports the results of the data-driven prediction of the SINR outage probability. We can find that the predicted SINR outage probability matches well with the simulated.

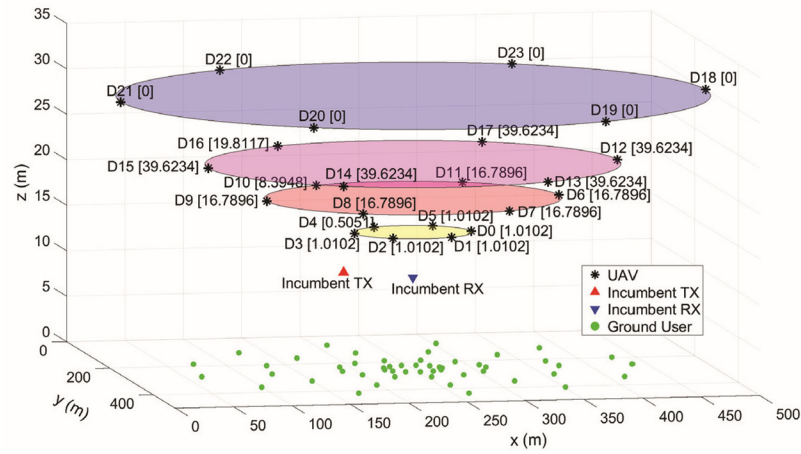


Figure 30: Case study of power control based on SwarmShare. D#1[#2]: #1 is the UAV index, and #2 denotes the transmission power of the UAV in mW.

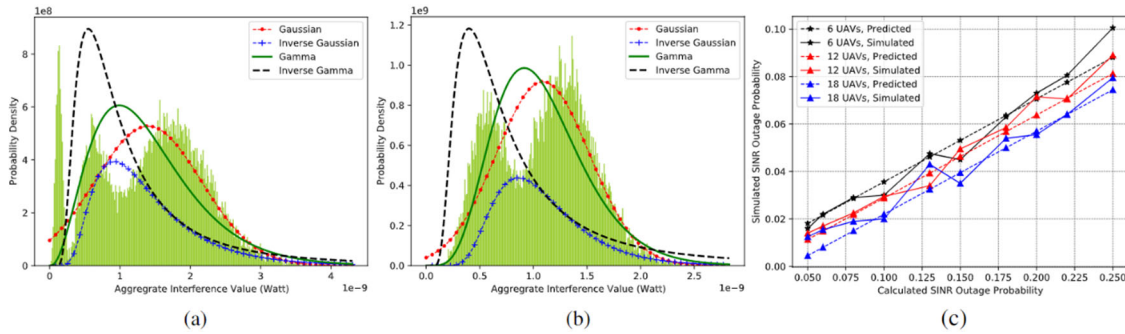


Figure 31: Aggregate interference pdf with (a) 10 and (b) 20 UAVs; (c) Validation of data-driven predication of the SINR outage probability for the incumbent system.

Case Study. Figure 31 shows an example of the power control results based on *SwarmShare*. To visualize the effects of the power control principles, in this example, all the 24 UAVs are deployed uniformly along 4 circles with different altitudes and radiuses. From the figure, it can be seen that lower transmission powers have been allocated to UAVs along the lower circles. Also, because of the shorter distances from the incumbent receiver, lower transmission powers has been allocated to the UAVs of the first circle from the bottom, e.g., 1.0102 mW for UAV 1 (i.e., D1[1] in Fig. 31) against 16.7896 mW for UAV 7 and 39.6234 mW for UAV 13 along the second and third circles, respectively. Moreover, along the same circle UAVs more aligned with the incumbent receiver have been allocated lower transmission powers, e.g., 8.3948 mW for UAV 10 vs 39.6234 mW for UAV 9 along the second circle. Finally, we notice in this example that all the UAVs along the fourth (i.e., the highest) circle have been allocated zero transmission power because no users are associated with them based on the shortest-distance association strategy. This also conforms to the third power control principle, i.e., with the same distance and relative angle, higher altitudes result in lower transmission power because of higher probability of LoS transmissions. It is worth pointing

out that the power allocation results are determined by jointly considering the three basic principles. In the following experiments, we will further evaluate the effectiveness of *SwarmShare* on the cross-system interference control.

In Figs. 32 and 33, we plot the instantaneous capacity achievable by the incumbent system and the UAV networks with different numbers of UAVs. In Fig. 32(a), we consider 6 hovering UAVs, and the maximum tolerable SINR outage probability is set to 0.05 for the incumbent system. The achievable capacity is plotted for 1000 time slots. Results indicate that the interference constraint of the incumbent system can be very well fulfilled, with SINR outage probability of 0.032. Similar results can be obtained with 12 and 18 hovering UAVs in Figs. 32(b) and (c), with the SINR outage probabilities of 0.029 and 0.021, respectively.

Figure 33 shows the corresponding results with moving UAVs. In this experiment, the network area is divided into a set of three-dimensional rectangles each of $50 \times 50 \times 10$ m³. The trajectory of the UAVs are controlled with exploitation probability of 0.98. The same as in Fig. 32, the incumbent system's interference constraints can be satisfied in all the tested cases, with SINR outage probability of 0.023, 0.019 and 0.018 for 6, 12 and 18 UAVs, respectively, all below the maximum tolerable outage probability 0.05. This verifies the effectiveness of *SwarmShare* in cross-system interference control.

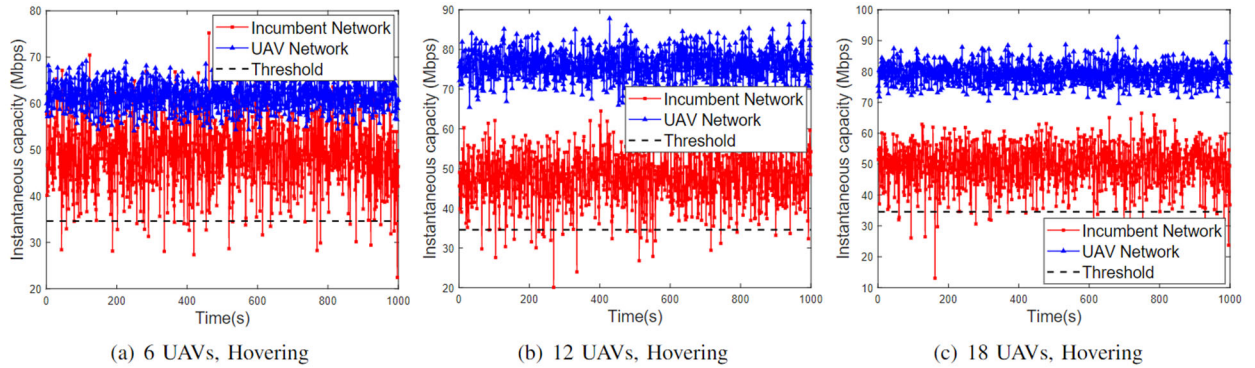


Figure 32: Instantaneous capacity of the incumbent system and the UAV network with hovering UAVs. The violation probabilities are (a) 0.032, (b) 0.029 and (c) 0.021, respectively.

Average Results. In Fig.34, we report the average capacity achievable by the incumbent system and the UAV network with the number of UAVs varying from 3 to 15 at steps of 3. Three UAV motility patterns are considered: i) random movement; ii) reinforcement learning controlled movement with exploitation probability of 0.98; and iii) hovering UAVs. The results are obtained by averaging over 50000 time slots for each mobility pattern. It can be seen that, as expected, an obvious capacity gain can be achieved by the UAV network with all the above three mobility patterns by deploying more UAVs. For example, for hovering UAVs,

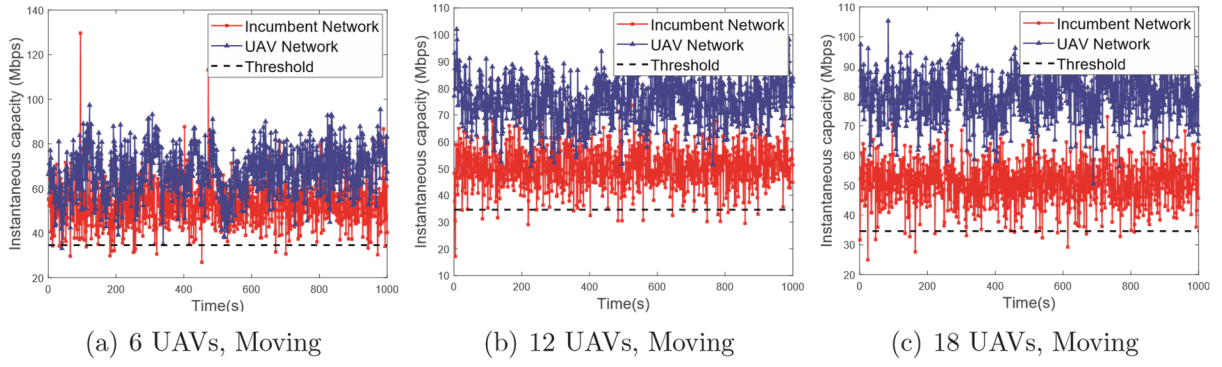


Figure 33: Instantaneous capacity of the incumbent system and the UAV network with moving UAVs. The violation probabilities are (a) 0.023, (b) 0.019 and (c) 0.018, respectively.

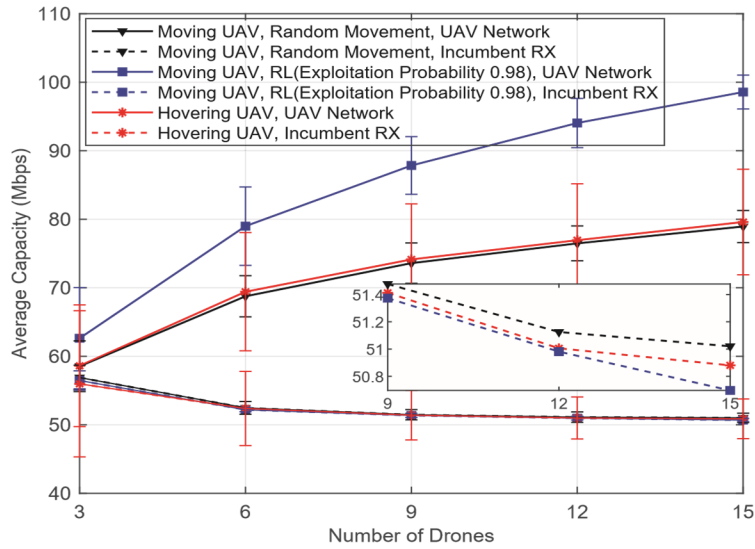


Figure 34: Average capacity of the incumbent system and UAV networks with moving UAVs.

the average capacity increases from around 60 Mbps with 3 UAVs to 80 Mbps with 15 UAVs. The capacity is further increased to around 100 Mbps with RL-controlled UAV movement. Particularly, we find that there is no obvious degradation in the capacity of the incumbent system when there are 6 or more coexisting UAVs. The average capacity of the incumbent system can be further increased with less UAVs, e.g., 3, because of the reduced cross-system interference.

In previous experiments (i.e., Fig. 32) the UAVs report their locations to the AFC in every time slot. In this experiment, we investigate the mobility resilience of *SwarmShare* for spectrum sharing in the presence of inaccurate UAV locations. The SINR outage probability results are reported in Fig. 35, where two mobility patterns are considered for the UAVs, i.e., random movement and RL-guided movement, and the maximum tolerable SINR outage probability is set to 0.05 for the incumbent system. The location reporting period is varied from 10 time slots to 60. It can be seen that the SINR outage probability of the incumbent

system increases monotonically with the location reporting period if the UAVs move in an uncontrolled manner, i.e., completely randomly. For example, the outage probability is around 0.07 when the reporting period is 10 time slots and can be up to 0.2 for 60 time slots.

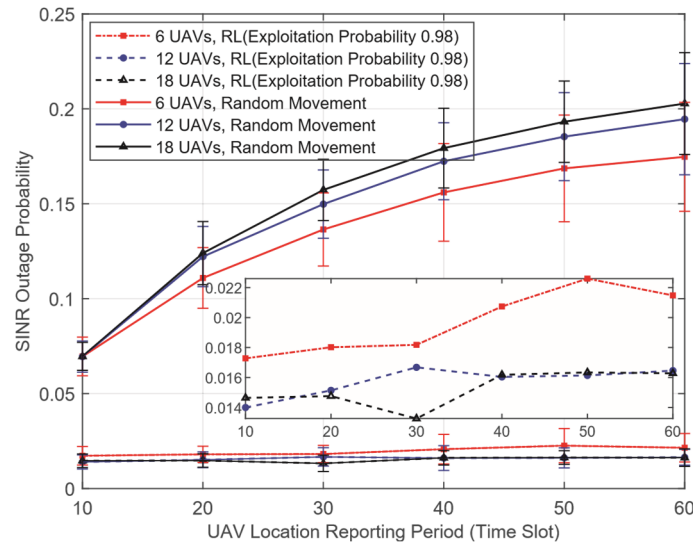


Figure 35: SINR outage probability vs UAV location reporting period.

In the case of controlled UAV movement, the SINR outage probability is barely affected by the location reporting period and always below the maximum tolerable. This is because the UAVs will stick with their current best locations at a high probability (0.98 in this experiment) while exploring new locations at a low probability (0.02). As a result, the topology of the UAV network and hence the statistical behavior of their aggregate interference to the incumbent system changes only slowly. Therefore, with controlled UAV movement, effective interference control can be achieved with *SwarmShare* in mobile scenarios even with inaccurate UAV locations, e.g., because of the temporary loss of the connections to the AFC.

3.6 RF-SITL Design

3.6.1 Background

The GE M-Series lab is built as a hardware-in-the-loop simulation environment for use with the GE M-Series computing platforms. GE M-100 are general computing platforms. They generally are used in Small UAVs (SUAVs) as flight control systems, with a combined mission-specific computing platform. For this project, the M-Series lab is configured with 3 M-100s as flight controllers for Simulated SUAVs. The three simulated SUAVs act as independent agents within an AFSIM simulation framework. AFSIM simulates and visualizes platforms and their interactions within a geographic context at both real-time and faster than real-time speeds. Platforms are vehicles (ground, air, space, surface, subsurface), buildings, or living

beings. Interactions include sensor detections, collisions, communications, etc., as well as weapons engagements. This project was to integrate University of Buffalo's RF-SITL into this environment to model RF communications between the three simulated SUAVs, as well as three ground agents defined wholly within the AFSIM environment.

3.6.2 Integration

To integrate RF-SITL into the M-Series Lab, various changes and adaptations were needed. To better understand the changes, it helps to understand the basic components to the existing RF-SITL as illustrated in

Figure 36. RF-SITL is implemented as a python script that programmatically defines a GNU Radio flowgraph that simulates NxN radio channels, with N denoting the number

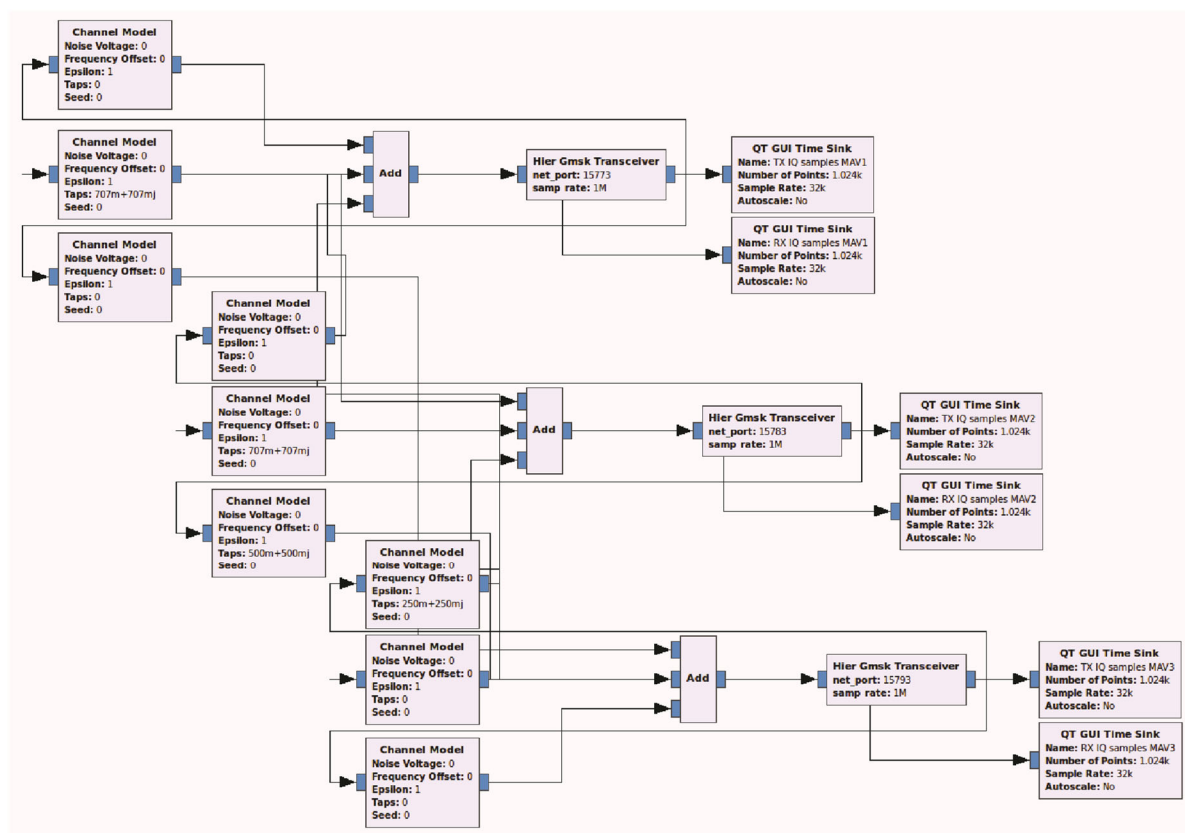


Figure 36: Example RF-SITL flowgraph with 3 transceivers and 3x3 channel blocks.

of radios being simulated. For each radio, there is a simulation flow that uses Channel Model Blocks for the RF signals between each simulated radio; these signals are summed together to create the simulated RF signal that is fed into a Hierarchical Transceiver Block for each radio. The Hierarchical Transceiver Block sends and receives the message via TCP ports, and the received messages are then output from the Hierarchical Transceiver Block and fed back into the Channel Model Blocks. Also instantiated in the RF-SITL flowgraph are GUI Time Sinks

for the RX and TX signals for each of the Hierarchal Transceiver Block for visualization of the radio signal.

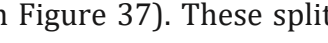
The final component is a Path Loss Model that receives the position for each radio and uses the distance between the radios to compute a matrix of channel taps that are used to configure the channel's impulse response through the Channel Model Blocks, simulating path loss and multi-path between radios.

Signal Sync. The original RF-SITL was dependent on a message for each radio being received, before processing the signal for each channel. This is problematic for the M-Series Lab as entities are not sending radio messages on a synchronized message schedule. To remedy this, RF-STIL was updated to allow for each radio to receive a null signal (0+0j) for other radios if no signal is being provided from the Transceiver Block.

Radio Position. To integrate RF-SITL, the Radio positions must be populated into RF-SITL from the M-Series Lab. Distributed Interactive Simulation (DIS) Entity Protocol Data Unit messages were chosen as they were already available from the M-Series Lab as the existing simulation's communication to AFSIM via DIS messages, and furthermore AFSIM has functionality for publishing DIS Entity Protocol Data Unit messages for the agents in AFSIM. This replaces existing messages that existed in RF-SITL. Functionally these messages serve the same purpose, however, they have slightly different message formats. DIS messages are typically sent via the User Datagram Protocol (UDP) rather than the Transmission Control Protocol (TCP) that is used by the UB-ANC messages. The open source OpenDIS python library was used to speed integration of the DIS messages into the RF-SITL Path Loss Model block.

Noise. To limit the range of a radio within the simulation, RF-SITL was modified to add a Noise Source Block. This block provides a Gaussian noise simulation which is added to the summation of the channel signal to add a noise floor to the radio signal, and the amplitude of the noise is modified to adjust the range of the radios. This noise amplitude was made externally controllable via a UDP port. This provides the ability to adjust the noise floor during execution.

Config File. To better control RF-SITL, a configuration file was added. This config file uses the JSON file format to define options for hierarchical radio selection, as well as options for send and receive ports associated with the message data. A further option was added for the default noise amplitude.

Flowgraph Splitting. The greatest change made to RF-SITL was to decompose the single RF-SITL flowgraph (like the one illustrated in  Figure 36 with 3 radios) into separate flowgraphs for each radio (like the one illustrated in  Figure 37). These split

flowgraphs are then able to run in parallel, greatly improving processing time by making use of multiple processing cores of the host environment. The flowgraphs use custom GNU radio blocks to share data, via UNIX sockets, with the other radio flowgraphs as well as a separate Path Loss Model component that processes the incoming DIS messages and computes the channel taps for all the flowgraphs.

A shell script is used to instantiate a flow graph for each simulated radio as well as the Path Loss Model. These changes are available in the GE-Integration branch of the RF-SITL github (<https://github.com/UBMCSL/gr-rfsitl>).

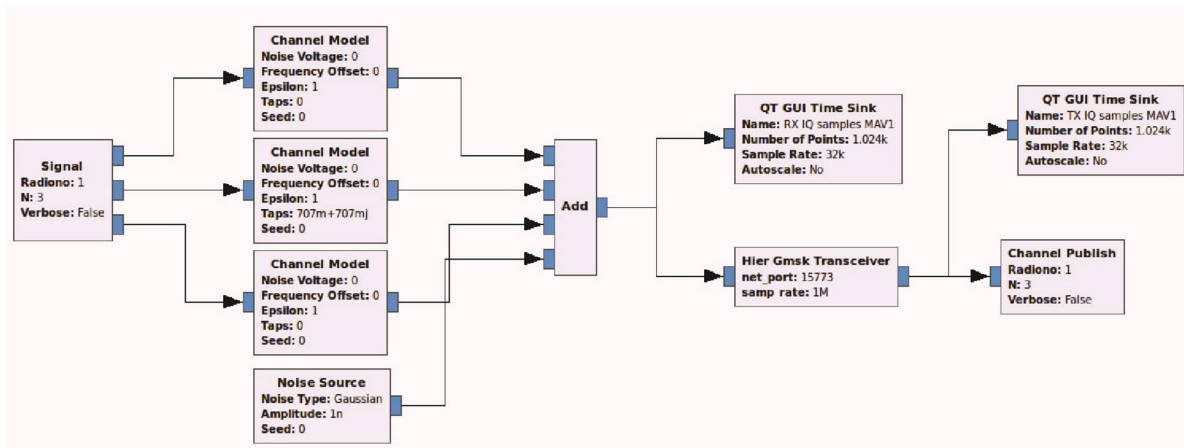


Figure 37: Example Single Radio Flowgraph in RF-SITL

3.6.3 Simulated COM Manager

To demonstrate the usage of RF-SITL, a COM manager was needed to produce and receive the radio traffic from each radio. The Simulated COM Manager sends/receives messages to the Transceiver Blocks via TCP. The simulated COM manager performs a few basic functions:

- 1) Send Message targeted to one or all radios.
- 2) Receive Messages from other radios, then send an Acknowledgement message back.
- 3) Route relay messages via a defined network topology.
- 4) Periodically send messages.

To see and manipulate the COM manager, a simple GUI was created that allows the control and display of the COM manager.

3.6.4 Simulation Results

We demonstrate the integration of RF-SITL into the M-Series lab through four progressively more complicated scenarios. Each scenario is executed using GMSK transceivers with a communication range set to 10 km. For each scenario, we present two figures showing: (1) a snapshot of the scenario in AFSIM; and (2) the packet transmissions between simulated nodes in the given scenario.

Scenario 1 is a simple simulated communication between two agents (BOB and FOB) defined within the AFSIM environment. The goal for this scenario is to show RF-SITL being integrated with static entities within AFSIM. This scenario is illustrated in Figure 38 with range rings around each entity for visual indication of the 10 km communication range. As illustrated in Figure 39, the successful execution of this scenario has messages being sent from BOB and received by FOB, followed by an acknowledgment message being sent from FOB and received by BOB. The total message round trip time was 1.4 s, including RF-SITL and COM Manager processing time.

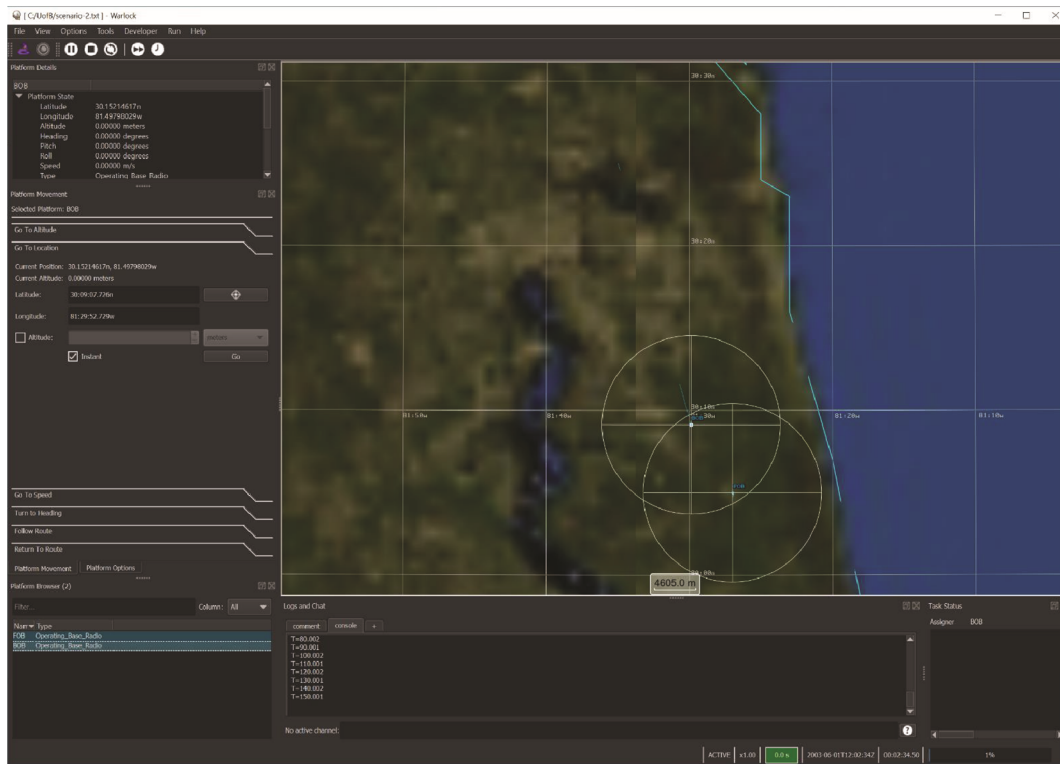


Figure 38: Scenario 1: Snapshot in AFSIM

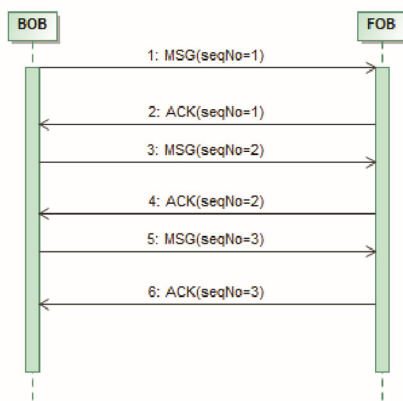


Figure 39: Scenario 1: Packet transmissions between nodes

Scenario 2 modifies Scenario 1 to space the AFSIM entities further apart and show that the communication link fails due to the distance-dependent channel models in RF-SITL. This scenario is illustrated in Figure 40. As illustrated in Figure 41, successful execution of this scenario has messages being sent from FOB but never received by BOB because they are too far apart.

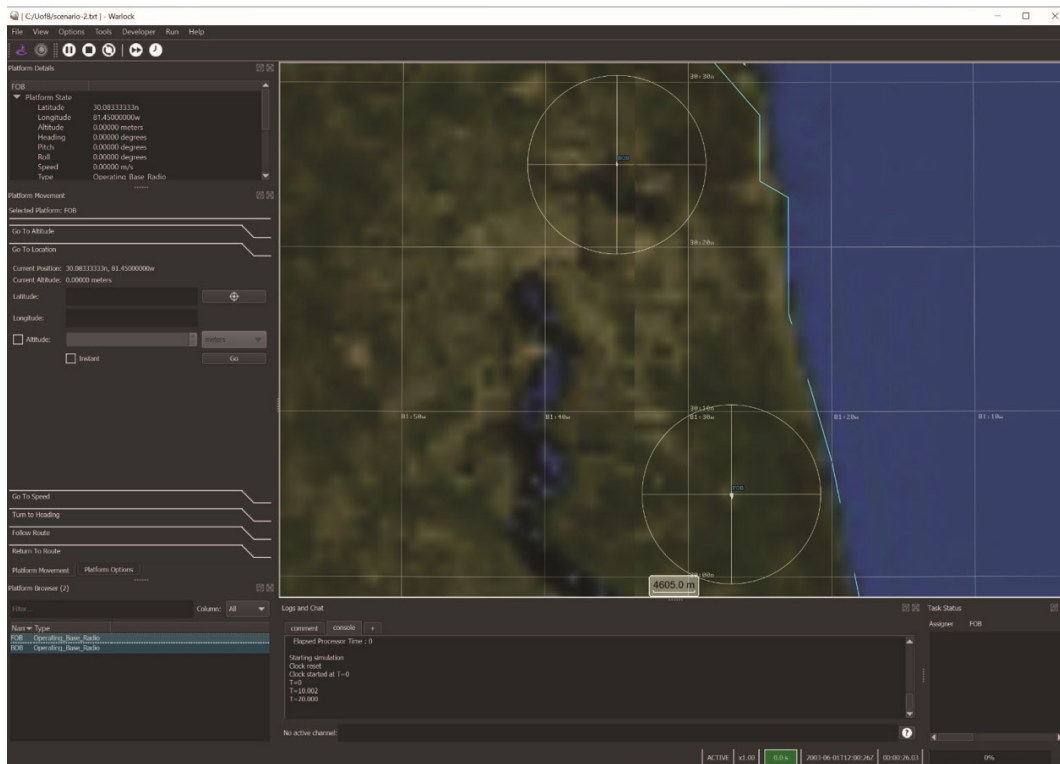


Figure 40: Scenario 2: Snapshot in AFSIM

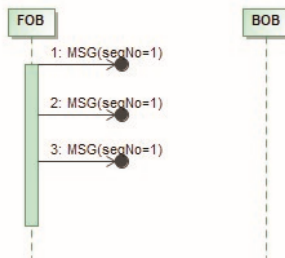


Figure 41: Scenario 2: Packet transmissions between nodes.

Scenario 3 further expands on scenario 2 by adding three UAVs to the simulation to act as communication relays between BOB and FOB. The three UAVs orbit in circles that are spaced equally between BOB and FOB. This scenario is illustrated in Figure 42. As illustrated in Figure 43, successful execution of this scenario has messages being sent from BOB, being relayed through SUAV-1, SUAV-2, and SUAV-3 before being received by FOB. FOB then sends an acknowledgment message back to BOB through the reverse route. The total round trip time was 4.7 s.

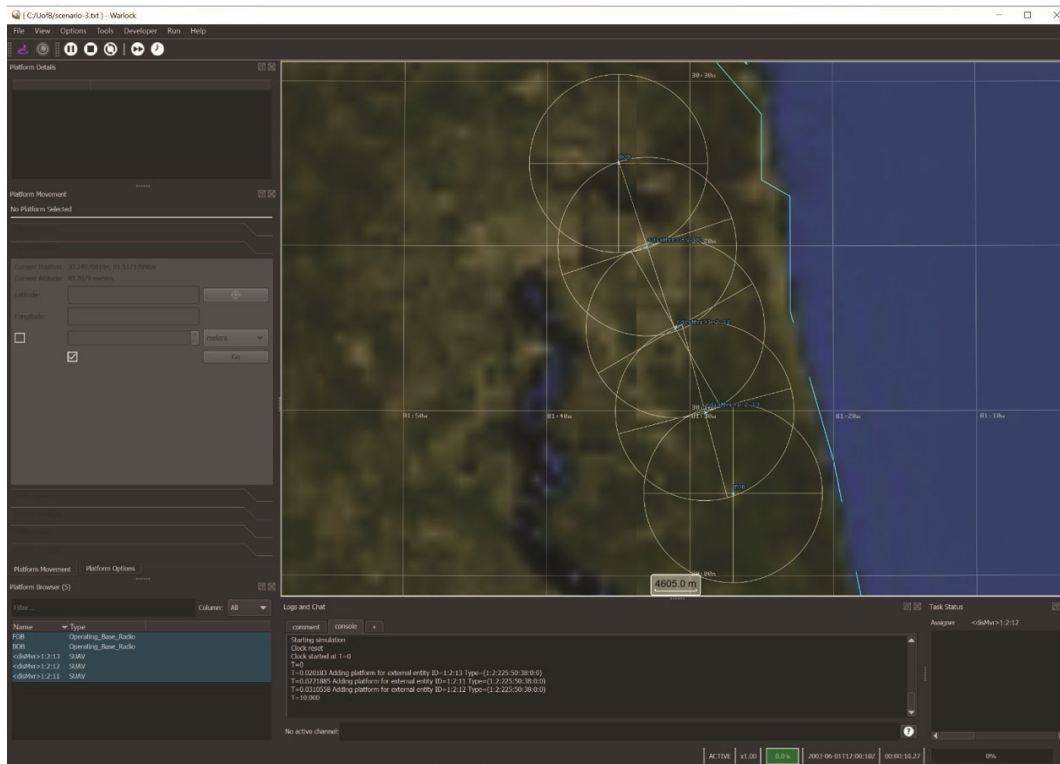


Figure 42: Scenario 3: Snapshot in AFSIM

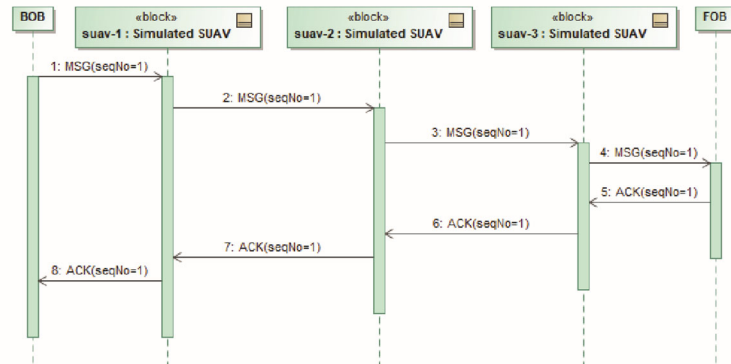


Figure 43: Scenario 3: Packet transmissions between nodes.

Scenario 4 further expands on scenario 3 by adding a third AFSIM agent (BA) that interferes with other transmissions. BA is positioned along the path from BOB to FOB, interrupting the communications between the relay UAVs. This scenario is illustrated in Figure 44, with the range ring for BA shown in red to indicate where the communications attack is coming from. As illustrated in Figure 45, successful execution of this scenario has messages being sent from BOB, being relayed to SUAV-1, and then failing to be delivered to SUAV-2 due to interference from BA.

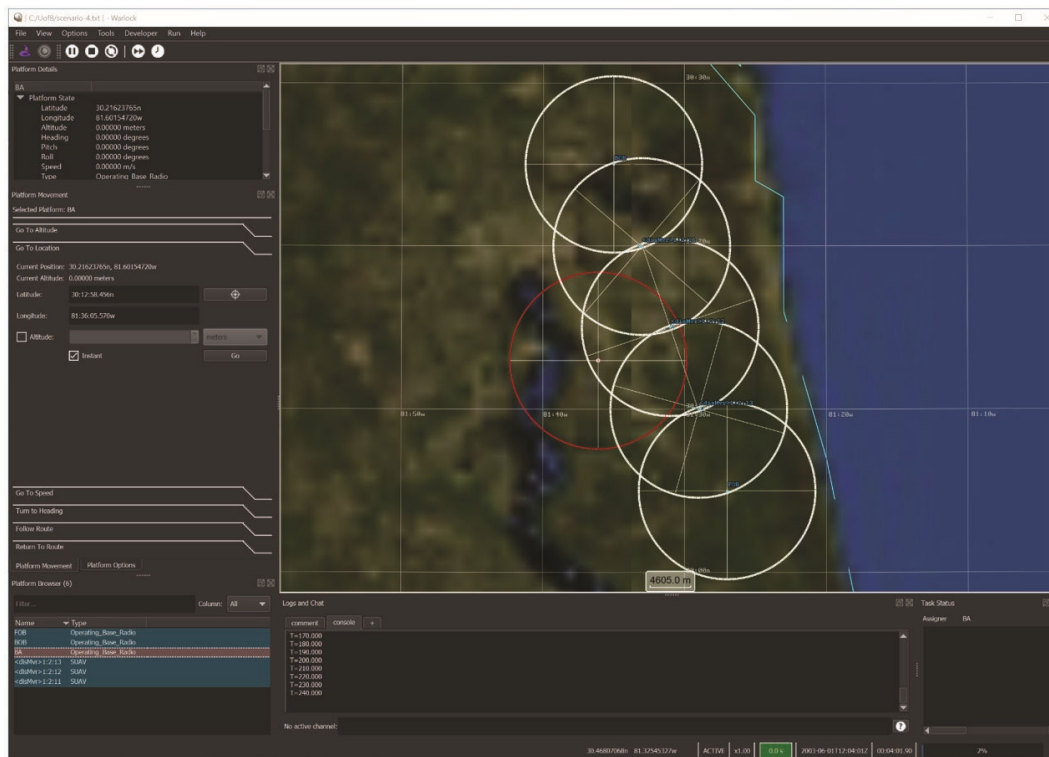


Figure 44: Scenario 4: Snapshot in AFSIM

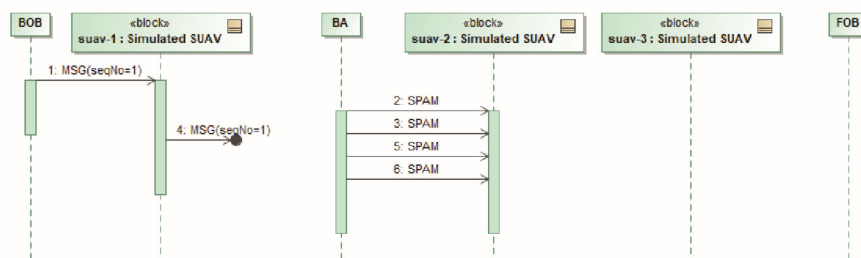


Figure 45: Scenario 4: Packet transmissions between nodes.

4 Results and Discussions

The project included numerous topics of research, so the results and discussions were folded into each topic discussion above (Protocol-Agnostic Packet Boundary Recognition (3.1.4 & 3.1.5), OSWireless Development (3.2.8, 3.2.9), CloudRAFT (3.3.4), FlyBeam (3.4.5), SwarmShare (3.5.5), RF-SITL Design (3.6.4)). This project has resulted in the following eleven (11) publications:

1. S. Krishna Moorthy, Z. Guan, N. Mastronarde, E. S. Bentley, M. Medley, “OSWireless: Hiding the Specification Complexity for Zero-Touch Software-Defined Wireless Networks,” submitted to *IEEE/ACM Transactions on Networking*.
2. S. Krishna Moorthy, A. Harindranath, Z. Guan, N. Mastronarde, E. S. Bentley, M. Medley, “Coordination Interface Design for Multi-Fidelity Flying Network Simulation with UBSim and NS-3,” submitted to *IEEE International Workshop on Wireless Communications and Networking in Extreme Environments (WCNEE) 2022*.
3. J. Hu, S. Krishna Moorthy, A. Harindranath, Z. Guan, N. Mastronarde, E. S. Bentley, and S. Pudlewski, “A Mobility-Resilient Spectrum Sharing Framework for Operating Wireless UAVs in the 6 GHz Band,” *IEEE/ACM Transactions on Networking*, under review, February 2022.
4. S. Krishna Moorthy, Z. Guan, N. Mastronarde, S. Pudlewski, and E. S. Bentley, “Distributed Beamforming for Swarm UAVs Through ESN Learning with Unknown Blockage Dynamics,” *IEEE/ACM Transactions on Networking*, under review, February 2022.
5. S. Krishna Moorthy, C. Lu, Z. Guan, N. Mastronarde, G. Sklivanitis, D. Pados, E. S. Bentley, and M. Medley, “CloudRAFT: A Cloud-based Framework for Remote Experimentation for Mobile Networks,” in *Proc. of IEEE International Workshop on Communication and Networking for Swarms Robotics (RoboCom)*, Virtual Conference, January 2022.
6. J. Hu, S. Krishna Moorthy, A. Harindranath, Z. Guan, N. Mastronarde, E. S. Bentley, and S. Pudlewski, “SwarmShare: Mobility-Resilient Spectrum Sharing for Swarm UAV Networking in the 6 GHz Band,” in *Proc. of IEEE International Conference on Sensing, Communication and Networking (SECON)*, Virtual Conference, July 2021.
7. S. Krishna Moorthy, Z. Guan, S. Pudlewski, E. S. Bentley, “FlyBeam: Echo State Learning for Joint Flight and Beamforming Control in Wireless UAV Networks,” in *Proc. of IEEE International Conference on Communications (ICC)*, Virtual/Montreal, Canada, 14-24 June, 2021.
8. M. McManus, Z. Guan, E. S. Bentley, S. Pudlewski, “Experimental Analysis of CrossLayer Sensing for Protocol-Agnostic Packet Boundary Recognition,” in *Proc. of IEEE INFOCOM Workshop on Wireless Sensor, Robot and UAV Networks (WiSARN)*, Virtual Conference, May 2021.
9. M. McManus, Z. Guan, N. Mastronarde, E. S. Bentley, “Accelerating Online Learning with Digital Twin for NextG Wireless Networks: Survey and Challenges,” to be submitted.

10. N. Mastronarde, D. Russell, Z. Guan, G. Sklivanitis, E. S. Bentley, M. Medley, “RFSITL: A Software-in-the-loop Channel Emulator for Swarm Networks”, to be submitted.
11. S. K. Moorthy, A. H., M. McManus, Z. Guan, N. Mastronarde, E. S. Bentley and M. Medley, “A Middleware for Digital Twin-Enabled Flying Network Simulations Using UBSim and UB-NC Airborne Networking and Communications testbed”, to be submitted for conference publication.

5 Conclusions

In this project we have designed new radio signal sensing and protocol classification techniques for automated discovery of the vulnerabilities of wireless systems, and simulated UAS networks in a CDO environment in the UB-ANC Emulator using SwarmControl for dynamic network management and control of UAS swarms. We have also integrated IQ-fidelity RF simulation into AFSIM to provide a complete common operating picture for swarm operations. The following six major accomplishments have been achieved through this project: i) we designed a new framework for protocol-agnostic sensing for over-the air (OTA) RF signals, by taking packet boundary recognition (PBR) as an example; ii) we designed OSWireless, a new control plane for optimizing software-defined wireless networks; iii) we proposed a cloud-based framework called CloudRAFT, based on which experimenters are allowed to remotely access and control experimental resources via public cloud AWS and share the resulting data and code via the cloud; iv) we proposed FlyBeam, a learning-based framework for joint flight and beamforming control in swarm UAV networks; v) we proposed *SwarmShare*, a mobility-resilient spectrum sharing framework for swarm UAV networking in the 6 GHz band; and vi) we collaborated with GE Aviation to integrate RF-SITL with their hardware-in-the-loop simulation environment comprising three M-100 flight controllers for simulated UAVs that act as independent agents within an AFSIM simulation framework. The research results have led to 11 publications.

6 References

- [1] F. Jin and T. Qiu, “Low Complexity Compressive Wideband Spectrum Sensing in Cognitive Radio,” in *Proc. of CITS*, Colmar, France, August 2018.
- [2] L. Zhang and Y. Liang, “Joint Spectrum Sensing and Packet Error Rate Optimization in Cognitive IoT,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7816–7827, October 2019.
- [3] L. Chen, “Security and Privacy on Physical Layer for Wireless Sensing: A Survey,” *Security and Privacy*, vol. 1, no. 5, pp. 1–10, April 2018.
- [4] Y. Zou, X. Wang, and L. Hanzo, “A Survey on Wireless Security: Technical Challenges, Recent Advances, and Future Trends,” *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, May 2016.
- [5] J. Du, H. Huang, X. J. Jing, and X. Chen, “Cyclostationary Feature Based Spectrum Sensing via Low-Rank and Sparse Decomposition in Cognitive Radio Network,” in *Proc. of ISCIT*, Qingdao, China, September 2016.
- [6] A. Ali and W. Hamouda, “Advances on Spectrum Sensing for Cognitive Radio Networks: Theory and Applications,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1277–1304, November 2017.
- [7] L. Zhang, Y. Liang, and M. Xiao, “Spectrum Sharing for Internet of Things: A Survey,” *IEEE Wireless Communications*, vol. 26, no. 3, pp. 132–139, June 2019.
- [8] M. K. Hanawal, Y. Hayel, and Q. Zhu, “Effective Utilization of Licensed and Unlicensed Spectrum in Large Scale Ad Hoc Networks,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 2, pp. 618–630, June 2020.
- [9] Q. M. Qadir, T. A. Rashid, N. K. Al-Salihi, B. Ismael, A. A. Kist, and Z. Zhang, “Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs,” *IEEE Access*, vol. 6, pp. 77454–77473, November 2018.
- [10] S. Chacko, M. D. Job *et al.*, “Security Mechanisms and Vulnerabilities in LPWAN,” in *Proc. of IOP Conference Series: Materials Science and Engineering*, Kerala State, India, April 2018.
- [11] S. Maru and T. X. Brown, “Packet Classification in Co-Mingled Traffic Streams,” in *Proc. of IEEE Workshop on Secure Network Protocols*, Princeton, New Jersey, USA, December 2009.
- [12] T. Dempsey, G. Sahin, Y. T. Morton, and C. M. Hopper, “Intelligent Sensing and Classification in Ad Hoc Networks: A Case Study,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 8, pp. 23–30, August 2009.
- [13] S. Wang, W. Guo, and M. D. McDonnell, “Downlink Interference Estimation Without Feedback for Heterogeneous Network Interference Avoidance,” in *Proc. of ICT*, Lisbon, Portugal, June 2014.

- [14] V. Bhardwaj, T. Goel, and G. Mahendru, "A Novel Wavelet Packet Based Direction of Arrival Estimation for Spectrum Sensing in Cognitive Radio Networks," in *Proc. of SPIN*, Noida, India, September 2018.
- [15] F. Liu, M. W. Marcellin, N. A. Goodman, and A. Bilgin, "Compressive Sampling for Detection of Frequency-Hopping Spread Spectrum Signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5513–5524, August 2016.
- [16] W. Xu, S. Wang, S. Yan, and J. He, "An Efficient Wideband Spectrum Sensing Algorithm for Unmanned Aerial Vehicle Communication Networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1768–1780, November 2018.
- [17] T. Datta, S. T. Anindo, and S. S. Alam, "Detection Performance Analysis for Wideband Cognitive Radio Network: A Compressive Sensing Approach," in *Proc. of IC4ME2*, Rajshahi, Bangladesh, July 2019.
- [18] K. Sherbin and V. Sindhu, "Cyclostationary Feature Detection for Spectrum Sensing in Cognitive Radio Network," in *Proc. of ICCS*, Madurai, India, May 2019.
- [19] B. Paul, A. R. Chiriyath, and D. W. Bliss, "Survey of RF Communications and Sensing Convergence Research," *IEEE Access*, vol. 5, pp. 252–270, December 2016.
- [20] C. Zhang, L. Yang, and X. Wang, "Discrete Wavelet Neural Network Group System for Digital Modulation Recognition," in *Proc. of IEEE ICCSN*, Xi'an, China, September 2011.
- [21] T. O'Shea, J. Corgan, and T. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Proc. of International conference on engineering applications of neural networks*, Aberdeen, United Kingdom, September 2016.
- [22] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-Air Deep Learning Based Radio Signal Classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, January 2018.
- [23] Q. Zhang, Z. Xu, and P. Zhang, "Modulation Recognition Using Wavelet-Assisted Convolutional Neural Network," in *Proc. of ATC*, Ho Chi Minh City, Vietnam, December 2018.
- [24] X. Li, F. Dong, S. Zhang, and W. Gui, "A Survey on Deep Learning Techniques in Wireless Signal Recognition," *Wireless Communications and Mobile Computing*, vol. 2019, February 2019.
- [25] A. K. C. Wong and M. You, "Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 5, pp. 599–609, September 1985.
- [26] A.-K. Seghouane, "A Kullback–Leibler Divergence Approach to Blind Image Restoration," *IEEE Transactions on Image Processing*, vol. 20, no. 7, pp. 2078–2083, January 2011.
- [27] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, March 2008.

- [28] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A Roadmap for Traffic Engineering in SDN-OpenFlow Networks," *Computer Network (Elsevier) Journal*, vol. 71, pp. 1–30, October 2014.
- [29] I. T. Haque and N. Abu-Ghazaleh, "Wireless Software Defined Networking: A Survey and Taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713– 2737, Fourthquarter 2016.
- [30] W. Wang, Y. Chen, Q. Zhang, and T. Jiang, "A Software-Defined Wireless Networking Enabled Spectrum Management Architecture," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 33–39, Jan. 2017.
- [31] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-Aware Controller Placement for Software-Defined Networks," in *Proc. of IFIP/IEEE International Symposium on Integrated Network Management*, Ghent, Belgium, May 2013.
- [32] M. Ambrosin, M. Conti, F. De G., and R. Poovendran, "LineSwitch: Tackling Control Plane Saturation Attacks in Software-Defined Networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1206–1219, April 2017.
- [33] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On Scalability of Software-Defined Networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, Feb. 2013.
- [34] A. Montazerolghaem, M. H. Y. Moghaddam, and A. Leon-Garcia, "OpenSIP: Toward Software-Defined SIP Networking," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 184–199, March 2018.
- [35] L. E. Li, Z. M. Mao, and J. Rexford, "Toward Software-Defined Cellular Networks," in *Proc. of European Workshop on Software Defined Networking (EWSDN)*, Darmstadt, Germany, Oct. 2012.
- [36] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A Low Cost Network Monitoring Framework for Software Defined Networks," in *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May 2014.
- [37] X. Ge, Z. Li, and S. Li, "5G Software Defined Vehicular Networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 87–93, July 2017.
- [38] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-Defined WSN Management System for IoT Applications," *IEEE Systems Journal*, vol. 2, no. 3, pp. 2074–2081, Sept. 2018.
- [39] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software Defined Radio Access Network," in *Proc. of HotSDN*, Hong Kong, China, August 2013.
- [40] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Ho"lzle, S. Stuart, and A. Vahdat, "B4: Experience With a Globally-Deployed Software Defined WAN," in *Proc. of the ACM SIGCOMM*, Hong Kong, China, Aug. 2013.

- [41] M. Jimenez and H. Kwok, "Building Express Backbone: Facebook's New Long-haul Network," 2017. [Online]. Available: <https://code.facebook.com/posts/1782709872057497/>
- [42] A. D. Ferguson, S. Gribble, C. Hong, C. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano, R. Alimi, S. S. Chen, M. Conley, S. Mandal, K. Nagaraj, K. N. Bollineni, A. Sabaa, S. Zhang, N. Zhu, and A. Vahdat, "Orion: Google's Software-Defined Networking Control Plane," in *Proc of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2021.
- [43] B. Cao, Y. Li, C. Wang, G. Feng, S. Qin, and Y. Zhou, "Resource Allocation in Software Defined Wireless Networks," *IEEE Network*, vol. 31, no. 1, pp. 44–51, Jan./Feb. 2017. [44] Open Networking Foundation, "OpenFlow Switch Specification," Oct. 2013.
- [45] D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN Control Plane for a Distributed Cloud-Edge Infrastructure: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 256–281, First Quarter 2021.
- [46] M. Priyadarsini and P. Bera, "Software Defined Networking Architecture, Traffic Management, Security, and Placement: A Survey," *Computer Networks (Elsevier)*, vol. 192, June 2021.
- [47] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures," *Proc. of IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.
- [48] G. Scutari, F. Facchinei, P. Song, D. P. Palomar, and J.-S. Pang, "Decomposition by Partial Linearization: Parallel Optimization of Multiuser Systems," *IEEE Trans. on Signal Process.*, vol. 63, no. 3, pp. 641–656, Feb. 2014.
- [49] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. USA: Now Publishers Inc, 2010.
- [50] K. Srivastava and A. Nedic, "Distributed Asynchronous Constrained Stochastic Optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, Aug. 2011.
- [51] G. Scutari and Y. Sun, "Parallel and Distributed Successive Convex Approximation Methods for Big-Data Optimization," *Lecture Notes in Mathematics, C.I.M.E, Springer Verlag series*, 2018.
- [52] <https://pypi.python.org/pypi/cogapp>.
- [53] D. P. Palomar and M. Chiang, "A Tutorial on Decomposition Methods for Network Utility Maximization," *IEEE JSAC*, vol. 24, no. 8, pp. 1439–1451, August 2006.
- [54] B. Johansson, P. Soldati, and M. Johansson, "Mathematical Decomposition Techniques for Distributed Cross-Layer Optimization of Data networks," *IEEE JSAC*, vol. 24, no. 8, pp. 1535–1547, August 2006.

- [55] Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, “WNOS: Enabling Principled Software-Defined Wireless Networking,” *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1391–1407, June 2021.
- [56] B. Buchberger, “Symbolic Computation (An Editorial),” *Journal of Symbolic Computation*, pp. 1–6, 1985.
- [57] “Simpy.” [Online]. Available: <https://pypi.org/project/simpy/>
- [58] D. Bertsekas and R. Gallager, *Data Networks*. USA: Prentice Hall, 2000.
- [59] Open RAN Alliance, “O-RAN: Towards an Open and Smart RAN,” *white paper*, October, 2018.
- [60] S. Niknam, A. Roy, H. S. Dhillon, S. Singh, R. Banerji, J. H. Reed, N. Saxena, and S. Yoon, “Intelligent O-RAN for Beyond 5G and 6G Wireless Networks,” *arXiv.org*, May 2020. [Online]. Available: <https://arxiv.org/pdf/2005.08374.pdf>
- [61] L. Bonati, M. Polese, S. D’Oro, S. Basagni, and T. Melodia, “Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead,” *Computer Networks*, vol. 182, pp. 1–28, Dec. 2020.
- [62] G. Hampel, D. Abush-Magder, A. Diaz, L. M. Drabeck, M. J. Flanagan, J. M. Graybeal, J. D. Hobby, M. MacDonald, P. A. Polakos, J. Srinivasan, H. Trickey, L. Zhang, and G. Rittenhouse, “The New Paradigm for Wireless Network Optimization: A Synergy of Automated Processes and Human Intervention,” *IEEE Communications Magazine*, vol. 43, no. 3, pp. S14–S21, March 2005.
- [63] L. S. Bai, R. P. Dick, P. H. Chou, and P. A. Dinda, “Automated Construction of Fast and Accurate System-Level Models For Wireless Sensor Networks,” in *Proc. of Design, Automation Test in Europe*, Grenoble, France, March 2011.
- [64] C. Di Martino, M. Cinque, and D. Cotroneo, “Automated Generation of Performance and Dependability Models for the Assessment of Wireless Sensor Networks,” *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 870–884, June 2012.
- [65] M. Ge, J. B. Hong, W. Guttman, and D. S. Kim, “A Framework for Automating Security Analysis of the Internet of Things,” *Journal of Network and Computer Applications*, vol. 83, pp. 12–27, April 2017.
- [66] K. Go˘karslan, “Menes: Towards a Generic, Fully-Automated Test and Validation Platform for Wireless Networks,” *arXiv:2006.02270*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.02270>
- [67] S. Harte, E. Popovici, B. O’Flynn, and C. O’Mathu’na, “THAWS: Automated Design and Deployment of Heterogeneous Wireless Sensor Networks,” *WSEAS Transactions on Circuits and Systems archive*, vol. 7, pp. 829–838, September 2008.
- [68] J. Beutel, M. Dyer, R. Lim, C. Plessl, M. Wohrle, M. Yucel, and L. Thiele, “Automated Wireless Sensor Network Testing,” in *Proc. of International Conference on Networked Sensing Systems*, Braunschweig, Germany, June 2007.

- [69] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," *arXiv:1606.04474*, 2016. [Online]. Available: <https://arxiv.org/abs/1606.04474>
- [70] S. Salman, C. Streiffer, H. Chen, T. Benson, and A. Kadav, "DeepConf: Automating Data Center Network Topologies Management with Machine Learning," in *Proc. of Workshop on Network Meets AI ML*, Budapest, Hungary, August 2018.
- [71] R. Shimizu, K. Tei, Y. Fukazawa, and S. Honiden, "Model Driven Development for Rapid Prototyping and Optimization of Wireless Sensor Network Applications," in *Proc. of Workshop on Software Engineering for Sensor Network Applications*, Waikiki, Honolulu, HI, May 2011.
- [72] H. Deng, Q. Li, Y. Li, S. Lu, C. Peng, T. Raza, Z. Tan, Z. Yuan, and Z. Zhang, "Towards Automated Intelligence in 5G Systems," in *Proc. of International Conference on Computer Communication and Networks (ICCCN)*, Vancouver, BC, August 2017.
- [73] Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, "WNOS: An Optimization-based Wireless Network Operating System," in *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Los Angeles, USA, June 2018.
- [74] L. Bonati, S. D'Oro, L. Bertizzolo, E. Demirors, Z. Guan, S. Basagni, and T. Melodia, "CellOS: Zero-touch Softwarized Open Cellular Networks," *Computer Networks (COMNET)*, vol. 180, Oct. 2020.
- [75] L. Bertizzolo, S. D'Oro, L. Ferranti, L. Bonati, E. Demirors, Z. Guan, T. Melodia, and S. Pudlewski, "SwarmControl: An Automated Distributed Control Framework for Self-Optimizing Drone Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada, July 2020.
- [76] "Platforms for Advanced Wireless Research (PAWR) Program." [Online]. Available: <https://advancedwireless.org/>
- [77] J. Breen, A. Buffmire, J. Duerig, K. Dutt, E. Eide, M. Hibler, D. Johnson, S. K. Kasera, E. Lewis, D. Maas, A. Orange, N. Patwari, D. Reading, R. Ricci, D. Schurig, L. B. Stoller, J. Van der Merwe, K. Webb, and G. Wong, "POWDER: Platform for Open Wireless Data-driven Experimental Research," in *Proc. of International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, London, United Kingdom, Sep. 2020.
- [78] D. Raychaudhuri, I. Seskar, G. Zussman, T. Korakis, D. Kilper, T. Chen, J. Kolodziejwski, M. Sherman, Z. Kostic, X. Gu, H. Krishnaswamy, S. Maheshwari, P. Skrimponis, and C. Gutterman, "Challenge: COSMOS: A City-Scale Programmable Testbed for Experimentation with Advanced Wireless," in *Proc. of International Conference on Mobile Computing and Networking*, London, United Kingdom, April 2020.
- [79] V. Marojevic, I. Guvenc, R. Dutta, M. L. Sichitiu, and B. A. Floyd, "Advanced Wireless for Unmanned Aerial Systems: 5G Standardization, Research Challenges, and AERPAW Architecture," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 22–30, June 2020.

- [80] "Agriculture and Rural Communities (ARA)." [Online]. Available: <https://arawireless.org/>
- [81] S. K. Moorthy and Z. Guan, "Beam Learning in MmWave/THz-band Drone Networks Under In-Flight Mobility Uncertainties," *IEEE Transactions on Mobile Computing*, accepted for publication, Oct. 2020.
- [82] S. K. Moorthy and Z. Guan, "FlyTera: Echo State Learning for Joint Access and Flight Control in THzenabled Drone Networks," in *Proc. of IEEE International Conference on Sensing, Communication and Networking (SECON)*, Como, Italy, June 2020.
- [83] Z. Ke, Z. Li, Z. Cao, and P. Liu, "Enhancing Transferability of Deep Reinforcement Learning-Based Variable Speed Limit Control Using Transfer Learning," *IEEE Transactions on Intelligent Transportation Systems*, accepted, May 2020.
- [84] "Amazon Web Services." [Online]. Available: <https://aws.amazon.com/>
- [85] "Ettus research." [Online]. Available: <https://www.ettus.com/>
- [86] "AWS Cognito." [Online]. Available: <https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazoncognito.html>
- [87] T. E. Anderson, M. D. Dahlin, J. M. Neefe, D. A. Patterson, D. S. Roselli, and R. Y. Wang, "Serverless Network File Systems," in *Proc. of ACM symposium on Operating systems principles*, Copper Mountain, Colorado, Dec. 1995.
- [88] G. Adzic and R. Chatley, "Serverless Computing: Economic and Architectural Impact," in *Proc. of Joint Meeting on Foundations of Software Engineering*, Paderborn, Germany, Aug. 2017.
- [89] "Fast Reverse Proxy (FRP)." [Online]. Available: <https://github.com/fatedier/frp>
- [90] T. Richardson and K. R. Wood, "The RFB Protocol," *ORL, Cambridge*, Jan. 1998.
- [91] "Configurable - Programmable Mecanum Wheel Vectoring Robot - IG52 DB." [Online]. Available: <https://www.superdroidrobots.com/shop/item.aspx/configurableprogrammable-mecanum-wheel-vectoring-robot-ig52-db/1788>.
- [92] "Super-Beacon." [Online]. Available: <https://marvelmind.com/product/superbeacon/>.
- [93] T. Heycke and L. Spitzer, "Screen Recordings as a Tool to Document Computer Assisted Data Collection Procedures: A Tutorial using the Open Broadcaster Software (OBS)," *Psychologica Belgica*, 2019.
- [94] A. Nurrohman and M. Abdurrohman, "High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP)," in *Proc. of IEEE International Conference on Information and Communication Technology (ICoICT)*, Bandung, Indonesia, May 2018.

- [95] V. Sharma and R. Kumar, “A Cooperative Network Framework for Multi-UAV Guided Ground Ad hoc Networks,” *Journal of Intelligent Robotic Systems*, vol. 77, pp. 629–652, Jan. 2015.
- [96] K. G. Panda, S. Das, D. Sen, and W. Arif, “Design and Deployment of UAV-Aided Post-Disaster Emergency Network,” *IEEE Access*, vol. 7, pp. 102985–102999, July 2019.
- [97] M. Mandi Azari et al., “Cellular UAV-to-UAV Communications,” in *Proc. of PIMRC*, Istanbul, Turkey, Sept. 2019.
- [98] Q. Wu and R. Zhang, “Common Throughput Maximization in UAV-Enabled OFDMA Systems With Delay Consideration,” *IEEE Transactions on Commun.*, vol. 66, no. 12, pp. 6614–6627, Nov. 2018.
- [99] Q. Wu and Y. Zeng, “Joint Trajectory and Communication Design for Multi-UAV Enabled Wireless Networks,” *IEEE Transactions on Wireless Commun.*, vol. 17, no. 3, pp. 3417–3442, Nov. 2018.
- [100] L. F. L. B. E. D. Z. G. T. M. L. Bertizzolo, S. D’Oro and S. Pudlewski, “SwarmControl: An Automated Distributed Control Framework for Self-Optimizing Drone Networks,” in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada, July 2020.
- [101] M. Mozaffari et al., “A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems,” *IEEE Commun. Surveys Tutorials*, vol. 21, no. 3, pp. 2334–2360, Nov. 2019.
- [102] A. Fotouhi et al., “Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges,” *IEEE Commun. Surveys Tutorials*, vol. 21, no. 4, pp. 2109–2121, Nov. 2019.
- [103] S. Mohanti et al., “AirBeam: Experimental Demonstration of Distributed Beamforming by a Swarm of UAVs,” in *Proc. of IEEE MASS*, Monterey, CA, USA, Nov. 2019.
- [104] Y. Lu, J. Fang, Z. Guo, and J. A. Zhang, “Distributed Transmit Beamforming for UAV to Base Communications,” *China Commun.*, vol. 16, no. 1, pp. 15–25, 2019.
- [105] S. Yeh, J. Chamberland, and G. H. Huff, “An Investigation of Geolocation-Aware Beamforming Algorithms for Swarming UAVs,” in *Proc. of IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting*, 2017, pp. 641–642.
- [106] P. Dinh, T. M. Nguyen, C. Assi, and W. Ajib, “Joint Beamforming and Location Optimization for Cooperative Content-Aware UAVs,” in *Proc. of IEEE WCNC*, Marrakesh, Morocco, Apr. 2019.
- [107] W. Yuan, C. Liu, F. Liu, S. Li, and D. W. K. Ng, “Learning-based Predictive Beamforming for UAV Communications with Jittering,” *IEEE Wireless Commun. Letters*, 2020.
- [108] W. Miao et al., “Position-Based Beamforming Design for UAV Communications in LTE Networks,” in *Proc. of IEEE ICC*, Shanghai, China, May 2019.

- [109] I. Kocaman, "Distributed beamforming in a swarm UAV network," *Calhoun: The NPS Institutional Archive DSpace Repository*, 2008. [Online]. Available: <https://calhoun.nps.edu/handle/10945/4215>
- [110] Z. Guan, N. Cen, T. Melodia, and S. Pudlewski, "Joint Power, Association and Flight Control for Massive-MIMO Self-Organizing Flying Drones," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1491–1505, August 2020.
- [111] D. C. Araujo et al., "Massive MIMO: Survey and Future Research Topics," *IET Commun.*, vol. 10, no. 15, pp. 1938–1946, Oct. 2016.
- [112] P. Zhang and J. Chen and X. Yang and N. Ma and Z. Zhang, "Recent Research on Massive MIMO Propagation Channels: A Survey," *IEEE Commun. Magazine*, vol. 56, no. 12, pp. 22–29, Dec. 2018.
- [113] K. Ntontin and C. Verikoukis, "Toward the Performance Enhancement of Microwave Cellular Networks Through THz Links," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 5635–5646, July 2017.
- [114] T. Bai, R. Vaze, and R. W. Heath, "Analysis of Blockage Effects on Urban Cellular Networks," *IEEE Transactions on Wireless Commun.*, vol. 13, no. 9, pp. 5070–5083, Sept. 2014.
- [115] "Spatial channel model for multiple input multiple output (MIMO) simulations (3GPP TR 25.996 version 11.0.0 release 11)." [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html>
- [116] J. Dang, C. Li, and Y. Meng, "Research on Least Square Channel Estimation Algorithm Based on Phase Compensation," in *Proc. of IEEE ICC*, Chengdu, China, Oct. 2016.
- [117] T. Parfait, Y. Kuang, and K. Jerry, "Performance analysis and comparison of ZF and MRT based downlink massive MIMO systems," in *Proc. of ICUFN*, Shanghai, China, July 2014.
- [118] M. Lukoševičius, "A Practical Guide to Applying Echo State Networks," *Lecture Notes*, Jan. 2012.
- [119] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction." Cambridge, MA, USA: MIT Press, 1998.
- [120] "Sequential Least Squares Programming (SLSQP)." [Online]. Available: <http://www.pyopt.org/reference/optimizers.slsqp.html>
- [121] Z. Chu, W. Hao, P. Xiao, and J. Shi, "UAV Assisted Spectrum Sharing Ultra-Reliable and Low-Latency Communications," in *Proc. of IEEE GLOBECOM*, Waikoloa, USA, Dec. 2019.
- [122] B. Shang, L. Liu, R. M. Rao, V. Marojevic, and J. H. Reed, "3D Spectrum Sharing for Hybrid D2D and UAV Networks," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5375–5389, May 2020.

- [123] “Unlicensed Use of 6 GHz Band,” in *Notice of Proposed Rulemaking ET Docket No.18295; GN Docket No. 17-183*, Oct. 2018.
- [124] V. Sathya, M. I. Rochman, M. Ghosh, and S. Roy, “Standardization Advances for Cellular and Wi-Fi Coexistence in the Unlicensed 5 and 6 GHz Bands,” *GetMobile*, vol. 24, no. 1, pp. 5–15, March 2020.
- [125] B. Wang, Y. Sun, Z. Sun, L. D. Nguyen, and T. Q. Duong, “UAV-Assisted Emergency Communications in Social IoT: A Dynamic Hypergraph Coloring Approach,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7663–7677, April 2020.
- [126] J. Lyu, Y. Zeng, and R. Zhang, “Spectrum Sharing and Cyclical Multiple Access in UAV-Aided Cellular Offloading,” in *Proc. of IEEE GLOBECOM*, Singapore, January 2018.
- [127] H. Wang, J. Wang, G. Ding, J. Chen, Y. Li, and Z. Han, “Spectrum Sharing Planning for Full-Duplex UAV Relaying Systems With Underlaid D2D Communications,” *IEEE JSAC*, vol. 36, no. 9, pp. 1986–1999, August 2018.
- [128] T. Q. Duong, L. D. Nguyen, H. D. Tuan, and L. Hanzo, “Learning-Aided Realtime Performance Optimisation of Cognitive UAV-Assisted Disaster Communication,” in *Proc. of IEEE GLOBECOM*, Waikoloa, USA, February 2019.
- [129] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, “Survey on UAV Cellular Communications: Practical Aspects, Standardization Advancements, Regulation, and Security Challenges,” *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3417–3442, March 2019.
- [130] M. Zareei, C. Vargas-Rosales, R. V. Hernandez, and E. Azpilicueta, “Efficient Transmission Power Control for Energy-harvesting Cognitive Radio Sensor Network,” in *Proc. of IEEE International Symposium on PIMRC Workshops*, Istanbul, Turkey, Sept. 2019.
- [131] Jie Xiang, Yan Zhang, and Tor Skeie, “Joint Admission and Power Control for Cognitive Radio Cellular Networks,” in *Proc. of IEEE Singapore Int’l. Conference on Communication Systems*, Guangzhou, China, Jan. 2008.
- [132] H. Guo, R. Long, and Y. Liang, “Cognitive Backscatter Network: A Spectrum Sharing Paradigm for Passive IoT,” *IEEE Wireless Communications Letters*, vol. 8, no. 5, pp. 1423–1426, Oct. 2019.
- [133] M. Labib, A. F. Martone, V. Marojevic, J. H. Reed, and A. I. Zaghloul, “A Stochastic Optimization Approach for Spectrum Sharing of Radar and LTE Systems,” *IEEE Access*, vol. 7, pp. 60814–60826, April 2019.
- [134] C. Aydogdu *et al.*, “RadChat: Spectrum Sharing for Automotive Radar Interference Mitigation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 416–429, December 2021.
- [135] A. F. Martone, K. A. Gallagher, and K. D. Sherbondy, “Joint Radar and Communication System Optimization for Spectrum Sharing,” in *Proc. of IEEE Radar Conference (RadarConf)*, Boston, USA, April 2019.

- [136] M. L. Attiah, A. A. M. Isa, Z. Zakaria, M. Abdulhameed, M. K. Mohsen, and I. Ali, "A Survey of mmWave User Association Mechanisms and Spectrum Sharing Approaches: An Overview, Open Issues and Challenges, Future Research Trends," *Wireless Networks*, vol. 26, no. 4, pp. 2487–2514, 2020.
- [137] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An Edge-Computing Based Architecture for Mobile Augmented Reality," *IEEE Network*, vol. 33, no. 4, pp. 162–169, January 2019.
- [138] J. Cui, Y. Liu, and A. Nallanathan, "Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks," *IEEE Trans. on Wireless Communications*, vol. 19, no. 2, pp. 729–743, August 2020.
- [139] L. Bertizzolo, E. Demirors, Z. Guan, and T. Melodia, "CoBeam: Beamforming-based Spectrum Sharing with Zero Cross-Technology Signaling for 5G Wireless Networks," in *Proc. of IEEE INFOCOM*, Toronto, Canada, July 2020.
- [140] Y. Li, H. Zhang, K. Long, S. Choi, and A. Nallanathan, "Resource Allocation for Optimizing Energy Efficiency in NOMA-based Fog UAV Wireless Networks," *IEEE Network*, vol. 34, no. 2, pp. 158–163, September 2020.
- [141] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the Curse of Dimensionality Using Decompositions of Incomplete Tensors: Tensor-based Scientific Computing in Big Data Analysis," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 71–79, August 2014.
- [142] K. T. Hemachandra and N. C. Beaulieu, "Novel Representations for the Equicorrelated Multivariate Non-Central Chi-Square Distribution and Applications to MIMO Systems in Correlated Rician Fading," *IEEE Trans. on Commun.*, vol. 59, no. 9, pp. 2349–2354, March 2011.
- [143] S. Kusaladharma and C. Tellambura, "Aggregate Interference Analysis for Underlay Cognitive Radio Networks," *IEEE Wireless Communications Letters*, vol. 1, no. 6, pp. 641–644, December 2012.
- [144] A. A. Khuwaja, Y. Chen, and G. Zheng, "Effect of User Mobility and Channel Fading on the Outage Performance of UAV Communications," *IEEE Wireless Commun. Letters*, vol. 9, no. 3, pp. 367–370, March 2020.
- [145] A. F. Schmidt and C. Finan, "Linear Regression and the Normality Assumption," *Journal of clinical epidemiology*, vol. 98, pp. 146–151, June 2018.
- [146] S. K. Moorthy and Z. Guan, "Beam Learning in MmWave/THz-band Drone Networks Under In-Flight Mobility Uncertainties," *IEEE Transactions on Mobile Computing*, accepted for publication, Oct. 2020.
- [147] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power Control Based on Deep Reinforcement Learning for Spectrum Sharing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 4209–4219, June 2020.

- [148] S. A. Al-Ahmed, M. Z. Shakir, and S. A. R. Zaidi, "Optimal 3D UAV Base Station Placement by Considering Autonomous Coverage Hole Detection, Wireless Backhaul and User Demand," *Journal of Commun. and Networks*, vol. 22, no. 6, pp. 467–475, December 2020.
- [149] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*. USA: John Wiley & Sons, 2013.
- [150] Z. Guan, T. Melodia, and G. Scutari, "To Transmit or Not to Transmit? Distributed Queueing Games for Infrastructureless Wireless Networks," *IEEE/ACM Trans. on Netw.*, vol. 24, no. 2, pp. 1153–1166, April 2016.
- [151] R. K. Ganti and M. Haenggi, "Interference in Ad Hoc Networks with General MotionInvariant Node Distributions," in *Proc. of IEEE Int'l. Symposium on Information Theory*, Toronto, Canada, July 2008.

7 List of Acronyms

8-PSK	8 Phase-Shift Keying
16QAM	16 Quadrature Amplitude Modulation
AERPAW	Aerial Experimentation Research Platform for Advanced Wireless
AFC	Automated Frequency Controller
AFSIM	Advanced Framework for Simulation, Integration, and Modeling
AI	Artificial intelligence
APIs	Application Programming interfaces
ARM	Algorithm Repository Module
ASS	Amazon's serverless service
AWGN	Additive White Gaussian Noise
AWS	Amazon Web Service
BPSK	Binary Phase Shift Keying
CDO	Contested, denied, and operationally
CellSDN	Software-defined cellular networks
CNNs	Convolutional neural networks
COSMOS	Cloud enhanced Open Software defined Mobile wireless testbed for city-scale deployment
CPU	Central Processing Unit
CSD	Cross-spectral density
CSI	Channel state information
CSMA	Carrier Sense Multiple Access
DED	Discrete Event Driver
DIS	Distributed Interactive Simulation
DoS	Denial-of-service
EM	Expression Model
ESN	Echo State Network
FCC	Federal Communications Commission
GEAS	General Electric Aviation Systems
GMSK	Gaussian Minimum Shift Keying
GPS	Global Positioning System
GUI	Graphical User Interface
HID	Horizontal Index
IoT	Internet of Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
JSON	Javascript Object Notation
KLD	Kullback-Leibler Divergence
LOS	line-of-sight
LPWAN	Low-power, wide area network
LTE	Long Term Evolution

MAC	Medium Access Control
MCC-SDWN	Mobile cloud computing based software defined wireless network
MIMO	Multiple Input Multiple Output
MINLP	Mixed integer nonlinear non-convex programming
ML	Machine learning
MR	Modulation recognition
MRT	Maximum Ratio Transmission
NCM	Network Configuration Module
NEM	Network Element Module
NextG	Next-generation
NLOS	Non-line-of-sight
NSF	National Science Foundation
OaaS	Optimization-as-a-Service
OBS	Open Broadcaster Software
OFDMA	Orthogonal frequency-division multiple access
ONF	Open Networking Foundation
O-RAN	Open Radio Access Network
OTA	Over-the air
PAWR	Platforms for Advanced Wireless Research
PBR	Packet boundary recognition
PDU	Power Distribution Units
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
REST	R-Epresentational State Transfer
RF	Radio-frequency
RFB	Remote Frame Buffer protocol
RF-SITL	Radio Frequency Software-in-the-Loop
RL	Reinforcement learning
RMSE	Root-mean-square error
RSA	Rivest–Shamir–Adleman
RTMP	Real-Time Messaging Protocol
SDN	Software-defined networking
SITL	Software-in-the-loop
SLSQP	Sequential Least Squares Programming
SM	Script Model
SNR	Signal-to-noise ratio
SoftRAN	Software defined radio access network
SSH	Secure Shell
SUAV	Small UAV
TCP	Transmission Control Protocol
TDMA	Time division multiple access

UAVs	Unmanned aerial vehicles
UB	University at Buffalo
UDP	User Datagram Protocol
UNB	Ultra-Narrowband
U-NII	Unlicensed National Information Infrastructure
URL	Uniform Resource Locator
USA	United States of America
VBEP	View-behavior element pair
VM	Virtual machine
VNC	Virtual Network Computing
WiNAS	Wireless Network Abstraction Specification
WNOS	Wireless Network Operating System
ZF	Zero Forcing